« jspau.pdf »

Programmieren mit JavaScript

Ein Lernprojekt
von Herbert Paukert
Version 24.b - 2024

Eigenverlag

www.paukert.at

Inhaltsverzeichnis

Programme von Teil 1 *09*

Programme von Teil 2 *65*

Programme von Teil 3 *113*

Programme von Teil 4 *169*

Programme von Teil 5 *231*

Programme von Teil 6 *273*

Programme von Teil 7 *331*

Hinweis: Die beiden mit (*) markierten Programme sind in diesem Lehrbuch nicht aufgelistet.
Sie sind jedoch im Editor „**htmledit.exe**“ gespeichert und dort aufrufbar (siehe nächste Seite).

Programme im Anhang *391*
(HTML und CSS)

Hinweis 1: Zur Erzeugung von HTML-Seiten kann ein Programm wie „**NotePad++**“ von Microsoft verwendet werden oder das Programm „**htmledit.exe**“ des Autors. Dieses Windows-Programm verfügt über einen einfachen Texteditor mit dem HTML-, CSS- und JavaScript-Befehle geschrieben, in eine HTML-Datei abgespeichert und dann auch ausgeführt werden können. Dabei muss im Dateikopf der mitteleuropäische Zeichensatz <meta charset = „ISO-8859-1“> angegeben werden. Eine Umwandlung in den universellen Unicode-Zeichensatz „UTF-8“ ist möglich.

In „**htmledit.exe**“ sind alle 170 Programme des Lehrbuchs und noch zusätzliche 100 Programme gespeichert. Diese können in einen Ordner ausgelagert, editiert und ausgeführt werden.

Hinweis 2: Eine weit verbreitete Art der Überprüfung und Überwachung eines Programmcodes ist die Verwendung des so genannten **Console-Objektes**, welches in allen Browsern vorhanden ist und über deren Eigenschafts-Menü geöffnet werden kann (sehr oft auch mit der Funktionstaste **<F12>**). Mit dem Befehl **console.log(*Variablenliste*)** werden in Systemmeldungen die jeweiligen Werte der Variablen und auch mögliche Fehler angezeigt. Im ersten Teil des Lehrbuches werden die Ergebnisse hauptsächlich mit dem Meldungsfenster **alert()** ausgegeben. Diese Ausgaben können aber auch ohne Schwierigkeit mittels **console.log()** erfolgen.

Hinweis 3: Alle Programme im Buch sind im reinen (**pure**) JavaScript-Code geschrieben.

Wien, Version 24.b (im Mai 2024)

Programmieren mit JavaScript, Teil 1
Allgemeine SPRACH-Grundlagen

[1.1] Was ist ein Programm?

Am Anfang steht immer ein Problem. Dieses wird vom Programmierer analysiert und ein Lösungsverfahren (Algorithmus) entwickelt. Der Algorithmus wird sodann mit Hilfe einer Programmiersprache in einer Folge von entsprechenden Befehlen formuliert. Die Befehle bilden das Programm und steuern den Computer derart, dass die Problemlösung realisiert wird. Ein Programm enthält daher sowohl die notwendigen Daten als auch die entsprechenden Befehle zur Verarbeitung dieser Daten.

Programm:
*Daten (**Was** wird verarbeitet)*
*Befehle (**Wie** wird verarbeitet)*

Der Vorteil der Programmierung in einer höheren Computersprache ist die symbolische Codierung. Der Zugriff auf die Daten erfolgt mittels einer symbolischen Adressierung. D.h., der Programmierer kann die Daten über Variablenbezeichner (das sind beliebig wählbare Namen) ansprechen und braucht sich nicht um deren effektive Speicheradressen kümmern. Auch die Befehle werden durch symbolische Schlüsselworte dargestellt und entsprechen meist sehr umfangreichen internen Maschinencodes. Einfache symbolische Adressierung der Daten und mächtige symbolische Befehlsworte kennzeichnen somit eine Hochsprache, die dadurch relativ maschinenunabhängig und komfortabel wird. Damit zwischen Daten und Befehlen keine Verwechslungen auftreten, werden sie in getrennten Bereichen des Hauptspeichers abgelegt.

Vor der Verwendung von Daten in einem Programm muss deren Speicherformat festgelegt werden, damit der Computer den Daten entsprechende Bytes des Speichers zuordnen kann. Will man zum Beispiel eine Berechnung durchführen und das Ergebnis in der Variablen *Resultat* abspeichern, so muss dem Computer am Beginn des Programms mitgeteilt werden, dass die Variable *Resultat* eine reelle Zahl enthalten soll. Dies geschieht im **Definitionsteil** des Programms.

Variable und Konstante (Daten, deren Werte sich während der Laufzeit des Programmes nicht ändern) werden erzeugt, um damit Berechnungen durchzuführen. Zu diesem Zweck muss eine Menge von zulässigen Operationen zur Verfügung stehen (z.B. die Addition "+"). Aus primitiven Operationen werden mächtige Operationsabläufe zusammengesetzt. Dies erfolgt im **Ablaufteil** des Programmes. Zwei grundlegende Programmbefehle sind Wertzuweisung und Wertevergleich:

Wertzuweisung: Einer Variablen (***X***) wird der Wert einer anderen Variablen oder Konstanten (***Y***) oder ein Operationsergebnis zugewiesen, z.B. ***X = Y*** oder ***X = Y + Z*** .

Wertevergleich: Test auf Gleichheit zweier Variablen, z.B. ***if (X == Y)***. Dabei wird immer der linke Wert mit dem rechten Wert auf Übereinstimmung verglichen.

Das Schreiben der Programmbefehle (Quelltext) erfolgt mit einem komfortablen **Editor**. Die Umwandlung des symbolischen Codes in den maschinenverständlichen Binärcode übernimmt ein eigenes Übersetzungsmodul, entweder ein **Interpreter** oder ein **Compiler**. Zusätzlich werden den symbolischen Variablen reale Speicherplätze (Adressen) zugeordnet. Während ein Interpreter die einzelnen Befehle schrittweise ausführt, werden durch einen Compiler alle Programmbefehle in ein als Ganzes ausführbares Programm umgewandelt.

- **Ein Schubladen-Modell des Speichers**

Einer Variablen sind folgende drei Bestimmungen zugeordnet:

Name: Durch diesen wird der Speicherplatz adressiert (d.h. wo die Variable gespeichert ist).
Typ: Dieser definiert die Struktur des Speicherplatzes (d.h. wie die Bits angeordnet sind).
Wert: Der Inhalt des Speicherplatzes (d.h. Auswertung der Bits, z.B. als Zahl oder Zeichen).

Vereinfacht kann eine solche Variable als eine mit einem Namensschild versehene Schublade im Speicherkasten des Computers aufgefasst werden. Damit laufen in vereinfachter Weise bei einer **Wertzuweisung** ***X = Y + Z*** folgende Arbeitsschritte im Computer ab:

(1) Suche im Speicherkasten die Schublade ***Y***.
(2) Transportiere ihren Inhalt in ein Register in der Zentrale.
(3) Suche im Speicherkasten die Schublade ***Z***.
(4) Transportiere deren Inhalt in ein Register in der Zentrale.
(5) Transportiere die Registerinhalte ins Rechenwerk und führe dort den Additionsbefehl (+) aus.
(6) Stelle dann das Ergebnis in ein zentrales Register zurück.
(7) Transportiere das Rechenergebnis aus dem zentralen Register in die Speicher-Schublade ***X***.

Ein **Wertevergleich** ***if (X == Y)*** kann in vereinfachter Weise folgendermaßen dargestellt werden:

(1) Suche im Speicherkasten die Schublade ***Y***.
(2) Transportiere ihren Inhalt in ein zentrales Speicherregister.
(3) Suche im Speicherkasten die Schublade ***X***.
(4) Transportiere ihren Inhalt in ein zentrales Speicherregister.
(5) Transportiere diese zwei Registerinhalte in das Rechenwerk und vergleiche sie dort.
Setze, je nach Ausgang des Vergleiches (gleich, ungleich), ein Statusbit (1 = true, 0 = false) im Statusregister der Zentrale.
(6) Das Vergleichsergebnis kann dann vom Programm im zentralen Statusregister abgelesen und zur Steuerung des weiteren Programmablaufes verwendet werden.

Die symbolischen Programmbefehle werden im Hauptspeicher im maschinenverständlichen Binärcode abgelegt und von der Zentrale des Computers (CPU) hintereinander abgearbeitet. Dabei erfolgt die Durchführung eines einzelnen Maschinenbefehls in einem dreiteiligen Zyklus: Befehl holen (vom Hauptspeicher in die Zentrale), Befehl decodieren und Befehl ausführen. Die Arbeitsgeschwindigkeit dieses Maschinenzyklus hängt weitgehend von der Taktrate der CPU ab.

- **Ein Programmbeispiel**

Eine Liste von Personennamen wird über die Tastatur in einen dafür reservierten Bereich des Hauptspeichers eingegeben. Dort erfolgt eine alphabetische Sortierung. Zum Schluss wird die sortierte Liste am Drucker ausgegeben.

Der eigentliche Kern des Programmes liegt in der Entwicklung eines geeigneten Sortierverfahrens. Dazu gibt es verschiedene Möglichkeiten: Beispielsweise durchläuft man mehrmals schrittweise die eingespeicherte Liste und vergleicht jedes Listenelement mit seinem Nachfolger. Steht das Element im Alphabet hinter seinem Nachfolger, dann müssen die beiden Elemente in der Liste ihre Plätze tauschen. Verglichen werden dabei die ANSI-Codes der einzelnen Textzeichen der Listenelemente. Am Ende des Verfahrens erhält man eine sortierte Liste.

Das Programm zur Lösung des Problems könnte in folgende Teilschritte (Module) zerlegt werden:

→ Programmbeginn.
→ Reservierung eines Speicherbereichs von Textvariablen (Strings) für die Personennamen.
→ Tastatureingabe der Personennamen und deren Abspeicherung auf die vorher reservierten Textvariablen.
→ Sortierung der Variablenliste im Hauptspeicher mit einem entsprechenden Sortierverfahren. Dieses wird als Unterprogramm im eigentlichen Hauptprogramm erstellt.
→ Ausgabe der Textvariablen am Drucker.
→ Programmende.

- **Das Problem und seine Lösung**

Wir wollen nun ganz allgemein untersuchen, was man unter einem Problem und seiner Lösung versteht. Zur Illustration ein einfaches Beispiel: Eine sortierte Liste von ganzen Zahlen soll vorliegen. Eine neue Zahl soll in diese sortierte Liste an der richtigen Stelle eingefügt werden. Zur Erledigung dieser Aufgabe stellen wir zunächst die Zahl an das Ende der Liste. Dann vergleichen wir – beginnend mit dem letzten Listenelement – jedes Element mit seinem Vorgänger. Ist das Element kleiner als sein direkter Vorgänger, so tauschen die beiden Platz. Dadurch ist das Element um eine Stelle nach vor gerückt. Dieses Verfahren wird dann abgebrochen, wenn das Element größer oder gleich seinem Vorgänger ist. Dann ist die neue Zahl an der richtigen Stelle eingefügt.

Unser Beispiel demonstriert sehr anschaulich das allgemeine Schema von Problemlösungen:

[1] Es existiert ein unerwünschter Anfangszustand ***AZ***.

[2] Der Problemlöser hat einen erwünschten Zielzustand ***ZZ*** vor Augen.

[3] Der Problemlöser sucht nach einer Transformation, die aus einer Folge von Operationen besteht, welche den Anfangszustand über Zwischenstufen in den Zielzustand überführen (***AZ → ZZ***).

Meistens ist leider die Problemlösung nicht sofort einsehbar (evident), sondern sie wird durch verschiedene Barrieren erschwert. Beispielsweise könnte der Zielzustand zu grob und zu wenig präzise formuliert sein oder es stehen keine zielführenden Operationen zur Verfügung.

Die Entwicklung eines Programmes zur Lösung von Problemen erfolgt in bestimmten Schritten:

PROBLEMSTELLUNG

LÖSUNGSENTWURF

PROGRAMMIERUNG

PROGRAMMTESTUNG

Der wesentliche Schritt dabei ist der Lösungsentwurf (Programmentwurf).
Er gliedert sich in zwei Abschnitte:

→ **Analyse des Problems**

Analyse der Ausgabedaten (Was will ich erreichen?)
Analyse der Eingabedaten (Was ist vorgegeben?)
Analyse des Lösungsverfahrens (Algorithmus: Wie erreiche ich das Ziel?)

→ **Darstellung des Lösungsverfahrens**

In umgangsprachlicher Form
Als Struktogramm
Als Flussdiagramm

Die wichtigsten Richtlinien für einen strukturierten Programmentwurf sind der ***Top-Down-Entwurf*** und die ***Modularisierung***. Darunter versteht man einerseits die schrittweise Verfeinerung der Problemlösung (vom Groben zum Feinen) und andererseits die Gliederung in verschiedene, wohldefinierte Teilbereiche (Module).

[1.2] Web-Browser und JavaScript (JS)

Web-Browser sind Computerprogramme zur Darstellung von multimedialen Daten (Texte, Grafiken, Sounds, Videos) im Internet (World Wide Web, WWW). Grundsätzlich besitzen alle Web-Browser eine definierte Startseite und eine Adressleiste zur Eingabe von Internetadressen (URLs). Zusätzlich verfügen Browser über Schaltflächen zum Navigieren im Internet und zum Download von Dateien. Die 2018 häufig verwendeten Browser sind Google-Chrome (62%), Mozilla-Firefox (11%), Microsoft-Edge (5%), Apple-Safari (4%). Alle Browser verstehen folgende drei Sprachen: **HTML** (Hyper Text Markup Language), **CSS** (Cascading Style Sheets), **JS** (JavaScript). Mit HTML können Internetseiten strukturiert erzeugt werden. Mit CSS können Internetseiten reichhaltig ausgestaltet und präsentiert werden. Mit JavaScript wird die Interaktion der Internetseite mit ihrem jeweiligen Benutzer programmiert. JavaScript ist eine Sprache mit der Befehle für Web-Browser programmiert werden. Diese Befehle müssen in den HTML-Code einer Internetseite eingebunden werden. Damit das Ganze auch ausgeführt werden kann, wurde die so genannte JavaScript-Engine entwickelt, die eigentlich ein textbasierter Sprachinterpreter ist.

HLML- und JavaScript-Dokumente können am einfachsten mit einem Texteditor (z.B. „notepad++“ in Windows) geschrieben werden. Der JavaScript-Code wird von den Tags <**script**> und </**script**> eingeschlossen, entweder im Header-Teil oder im Body-Teil des HTML-Dokumentes. Als Alternative kann der JavaScript-Code in eine einfache Texdatei (z.B. „myprog.js“) geschrieben und dann mit der Anweisung <**script src** = 'myprog.js'></**script**> in das HTML-Dokument eingefügt werden.

Das unten aufgelistete, erste JavaScript-Programm („js01.html“) besteht nur aus dem einzigen Befehl **alert(Text)**, welcher in einem Meldungsfenster den Text ausgibt. Jeder Text muss entweder zwischen einfachen Hochkommas (**'**) oder doppelten Hochkommas (**"**) stehen, und jeder JavaScript-Befehl muss mit einem Strichpunkt (**;**) beendet werden.

```
<!DOCTYPE html>
<html>
<head>
   <title> JS01 </title>
   <meta charset="ISO-8859-1">
   <meta name=”author” content=”Herbert Paukert”>
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%;}
   </style>
   <script>
      alert ('Hallo Welt !');  // erzeugt ein erstes Meldungsfenster
   </script>
</head>
<body>
   <p> Programm JS01: Ausgabe von Meldungen. </p>
   <script>
      alert ('Ich bin Programm JS01 ! ');  /* erzeugt ein zweites Meldungsfenster */
   </script>
</body>
</html>
```

Anmerkung 1: Grundsätzlich besteht die alternative Möglichkeit nach <script> als ersten Programmbefehl 'use strict' zu schreiben. Dadurch wird der so genannte „**Strict Mode**“ aktiviert. Im Gegensatz zum normalen Modus werden vom JavaScript-Interpreter still ignorierte Fehler sofort angezeigt, automatisch bestimmte Fehler behoben und der Code schneller ausgeführt. Das vorliegende Buch verzichtet auf diesen Modus.

Anmerkung 2: Kommentare können in einer Programmzeile immer rechts von einem doppelten Schrägstrich geschrieben (// Kommentar) oder durch Spezialzeichen begrenzt (/* Kommentar */) werden.

Anmerkung 3: JavaScript ist eine case-sensitive Programmiersprache, d.h. bei Befehlsnamen und Variablennamen wird genau zwischen Klein- und Großschreibung unterschieden.

Anmerkung 4: Im **ersten Teil** des Lehrbuches wird im JavaScript-Code zur Vereinfachung kein Bezug genommen zu den Objekten im HTML-Dokument (z.B. Textfelder oder Schaltflächen). Auf die Verwendung von HTML-Objekten (Document Object Model, DOM) wird **vorerst** verzichtet.

Anmerkung 5: Die wichtigsten HTML-Tags und CSS-Styles sind im **Anhang** des Buches beschrieben.

[1.3] Einfache Variablentypen

In diesem Kapitel sollen einfache Variablentypen und einige vordefinierte Auswertungsfunktionen besprochen werden. Der Datentypus einer Variablen muss **nicht explizit definiert** werden und jede Variable kann mit dem Wert eines jeden Datentyps überschrieben werden.

(1) Zeichenketten (string), *strVar* = ***'Herbert';***
(2) Zahlen (number), *numVar* = ***174.58;***
(3) Wahrheitswerte (boolesch), *logVar* = ***true (false);***

Jede Anweisung (Befehl) besteht aus einer Textzeile, die von einem Semikolon (;) beendet wird.
Die wählbaren Namen für Variable müssen sich von den JavaScript-Schlüsselwörtern unterscheiden.
Beispiele von JavaScript-Schlüsselwörtern: ***const, var, let, do, else, for, if, new, . . .***
Variable sollten immer mit dem Schlüsselwort ***var*** bzw. ***let*** initialisiert (deklariert) werden.
Konstante werden mit dem Schlüsselwort ***const*** initialisiert und können nicht verändert werden.

[1.3.1] Zeichenketten (string)

Jede Zeichenkette (string) muss von einfachen (') oder doppelten Hochkommas (") eingeschlossen werden. Ein Beispiel einer Variableninitialisierung: ***var strVar = 'Herbert';***
Der Operator + verkettet zwei Strings: ***strVar + ' Meier'*** ergibt ***'Herbert Meier';***

- Die einfachste Form der Dateneingabe ist die Anweisung ***strVar = prompt('text', strVar);***

Dabei wird in einem Meldungsfenster ein Wert eingegeben und dann der Variablen zugewiesen.
'text' ist ein Informationstext und der alte Wert von ***strVar*** wird zusätzlich angezeigt.

- Die Anweisung ***result =confirm(Fragetext);*** zeigt den Fragetext und zwei Schalter [**Ok**] und [**Abbrechen**]. Wird [**Ok**] angeklickt, ist die Variable ***result*** wahr (true), andernfalls falsch (false).

- Die einfachste Form der Datenausgabe ist die Anweisung ***alert(strVar);***

Dabei wird die Variable in einem Meldungsfenster ausgegeben.
Das Steuerzeichen ***'\n'*** bewirkt im ausgegebenen Text einen Zeilenvorschub.

- Mit ***document.write(strVar)*** wird direkt in ein Dokument geschrieben.

Dabei können mehrere Variable durch Beistriche getrennt verwendet werden,
und es sind auch HTML-Tags (z.B. ein Zeilenvorschub ***'
'***) möglich.

```
<!DOCTYPE html>
<html>
<head>
   <title> JS02 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>

<body>
   <p> Programm JS02: Eingabe und Ausgabe von Daten. </p>
   <script>
      var name = '???';
      name = prompt('Bitte Deinen Namen eingeben', name);
      var ausgabe = 'Hallo ' + name + '!  Wie geht es Dir?';

      // Datenausgabe in einem Meldungsfenster
      alert(ausgabe);

      // Datenausgabe direkt in das aktuelle Dokument
      document.write('Dein Name ist','<br>',name);
      alert('Weiter');

      // Datenausgabe in ein neues Fenster (msgWindow)
      var myWindow = window.open('', 'msgWindow', 'left=20, top=10, width=200, height=100');
      myWindow.document.write('Dein Name ist','<br>',name);
   </script>
</body>
</html>
```

Eine Zeichenkette (String) besteht aus aneinander gereihten Zeichen, welche intern nummeriert (indiziert) sind. Das erste Zeichen hat immer den Index 0, das zweite Zeichen den Index 1, usw. Die String-Variablen sind intern als Objekte mit bestimmten Eigenschaften und Methoden (vordefinierte Auswertungsfunktionen) abgespeichert. Eigenschaften und Methoden werden immer mit einem Punkt (.) an den Variablennamen angefügt. Einige Beispiele:

```
var r = 'a';
var s = 'Herbert';
var t = '';                                  // Leerstring (Nullstring)
t = r + s;                                   // liefert eine Stringverkettung (also 'aHerbert')
len = s.length;                              // liefert die Anzahl der Zeichen (also 7)
pos = s.indexOf('er') ;                      // liefert die erste Position von 'er' in s (also 1)
pos = 1; s = s.substr(0,pos) + 'a' + s.substr(pos+1);   // ändert 'Herbert' zu 'Harbert'
ch = s.charAt(0);                            // liefert das Zeichen (Character) 'H'
```

```
<!DOCTYPE html>
<html>
<head>
   <title> JS03 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>
<body>
   <p> Programm JS03: Zeichenketten (Strings). </p>
   <script>
      var tex1 = 'Text';
      var tex2 = 'verkettung';
      var tex3 = tex1 + tex2;

      var len  = tex3.length;
      alert(tex3 + '\n' + 'Textlänge = ' + len);
      pos  = tex3.indexOf('x');                          // liefert die Position 2
      alert(tex3 + '\n' + 'Position von "x" = ' + pos);
      pos  = tex3.indexOf('Z');                          // liefert die Position -1
      alert(tex3 + '\n' + 'Position von "Z" = ' + pos);
      var ch = text3.charAt(4);                          // liefert das Zeichen 'v'
      var tex4 = tex3.substring(0,4);
      alert(tex3 + '\n' + 'Die ersten 4 Zeichen sind: ' + tex4);
      tex4 = tex3.substring(tex3.length-1,tex3.length);
      alert(tex3 + '\n' + 'Das letzte Zeichen ist: ' + tex4);
      tex5 = tex1.substr(0,2) + 's' + tex1.substr(3);    // 'Text' wurde zu 'Test' geändert
      alert(tex1 + '  wurde verändert zu  ' + tex5);

      code = 'ABC'.charCodeAt(0); alert(code);           // liefert den Zeichencode 65
      ch = String.fromCharCode(65); alert(ch);           // liefert das Zeichen 'A'
   </script>
</body>
</html>
```

Examples of String-Methods (All string methods return a new value. They do not change the original variable.)

```
charAt()        // Returns the character at the specified index (position)
concat()        // Joins two or more strings, and returns a new joined strings
endsWith()      // Checks whether a string ends with specified string/characters
includes()      // Checks whether a string contains the specified string/characters
indexOf()       // Returns the position of the first found occurrence of a specified value in a string
lastIndexOf()   // Returns the position of the last found occurrence of a specified value in a string
replace()       // Searches a string for a specified value, or a regular expression, and returns a new string
                // where the specified values are replaced
search()        // Searches a string for a specified value, or regular expression, and returns the position of the match
slice()         // Extracts a part of a string and returns a new string
split()         // Splits a string into an array of substrings
startsWith()    // Checks whether a string begins with specified characters
substr()        // Extracts the characters from a string, beginning at a specified start position,
                // and through the specified number of character (or to the end of the string)
substring()     // Extracts the characters from a string, between two specified indices (or to the end of the string)
toLowerCase()   // Converts a string to lowercase letters
toString()      // Returns the value of a String object
toUpperCase()   // Converts a string to uppercase letters
trim()          // Removes whitespace from both ends of a string
```

[1.3.2] Zahlen (number)

Zahlen (number) sind immer so genannte 64-Bit-Gleitkommazahlen. Ganze Zahlen haben keine Nachkommastellen. Die Number-Variablen sind intern als Objekte mit bestimmten Eigenschaften und Methoden (vordefinierte Auswertungsfunktionen) abgespeichert. Eigenschaften und Methoden werden immer mit einem Punkt (.) an den Variablennamen angefügt. Einige Beispiele:

```
var numVar1 = 12345;
var numVar2 = 123.45;                  // Kein Komma sondern Dezimalpunkt !

var numVar3 = 1.23e4;                  // = 1.23*10 4 = 12300
var numVar4 = 1.23e-3;                 // = 1.23*10 -3 = 0.00123

var s = '12345';                       // String-Variable
var z;                                 // Zahlen-Variable
z = parseFloat (s);                    // Umwandlung eines Strings in eine Gleitkommazahl
z = 2 / 3;   // = 0.666666666666 …….
z = z.toFixed(4);                      // = 0.6667 (auf vier Nachkommastellen gerundet)

s = numVar1.toString();                // Umwandlung einer Zahl in einen String
s = numVar1.toString(2);               // Umwandlung einer Zahl in einen String (dual)
s = numVar1.toString(16);              // Umwandlung einer Zahl in einen String (hex)
z = parseInt (s);                      // Umwandlung eines Strings in eine Ganzzahl
z = parseInt (s,2);                    // Umwandlung eines Strings (dual)  in eine Ganzzahl
z = parseInt (s,16);                   // Umwandlung eines Strings (hex) in eine Ganzzahl
```

```
<!DOCTYPE html>
<html>
<head>
   <title> JS04 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>
<body>
   <p> Programm JS04: Zahlen (numbers). </p>
   <script>
      var x = 1;
      var y = 3;
      var z1 = x + y;
      var z2 = x / y;

      alert(x + ' + ' + y + ' = ' + z1);
      alert(x + ' : ' + y + ' = ' + z2);

      x = prompt('Erste Zahl eingeben',x);
      y = prompt('zweite Zahl eingeben',y);
      z1 = x + y;                                  // String-Verkettung
      alert(x + ' + ' + y + ' = ' + z1);

      z1 = parseFloat(x) + parseFloat(y);          // Zahlen-Addition
      alert(x + ' + ' + y + ' = ' + z1);

      z1 = 1*x + 1*y;                              // Zahlen-Addition
      alert(x + ' + ' + y + ' = ' + z1);
   </script>
</body>
</html>
```

Der ***prompt***-Befehl liefert immer Strings. Das Additionszeichen (+) führt dann zu einer String-Verkettung. Um mit Zahlen zu rechnen, müssen daher die Strings vorher in Zahlen umgewandelt werden. Das kann entweder mit der vordefinierten Funktion ***parseFloat*** durchgeführt werden, oder die Variablen werden einfach mit **1** multipliziert. Dabei wird der Variablentyp automatisch von String zu Number konvertiert.

Anmerkung:

Die Anweisung ***n = n + 1;*** kann ersetzt werden durch ***n++;*** (Inkrementoperator).
Die Anweisung ***n = n – 1;*** kann ersetzt werden durch ***n-- ;*** (Dekrementoperator).

Die nachfolgende Tabelle zeigt alle einfachen arithmetischen Operationen in JavaScript, die direkt mit numerischen Variablen ausgeführt werden können.

Operator	Bedeutung	Beispiel
+	Addition	a = b + c;
-	Subtraktion	a = b – c;
*	Multiplikation	a = b * c;
/	Division	a = b / c;
%	Modulo (Rest einer Division)	a = b % c;

Eine interessante JavaScript-Funktion ist ***eval(Textausdruck)***, welche versucht den Textausdruck auszuwerten und - falls möglich - ein Ergebnis zurückliefert. Als Textausdruck können alle einfachen arithmetischen Operationen und auch die Potenzfunktion (x ** n) eingegeben werden, beispielsweise liefert ***eval((2+1)**3)*** genau 27.

```
<!DOCTYPE html>
<html>
<head>
   <title> JS05 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%;}
   </style>
</head>
<body>
   <p> Programm JS05: Evaluierung von Formeln. </p>
   <script>
      var formel, ergebnis, ausgabe;
      formel = prompt('Bitte eine Formel eingeben');
      ergebnis = eval(formel);
      ausgabe = formel + ' = ' + ergebnis;
      alert(ausgabe);
   </script>
</body>
</html>
```

Sollen kompliziertere mathematische Funktionen ausgeführt werden, dann müssen das JavaScript-Objekt **MATH** und dessen Eigenschaften und Methoden verwendet werden.

Wichtige Eigenschaften (Konstante)

```
Math.E          // Returns Euler's number
Math.PI         // Returns PI
Math.LOG10E     // Returns base 10 logarithm of E
```

Wichtige Methoden (Funktionen)

```
Math.abs(x)         // Returns the absolute value of x
Math.acos(x)        // Returns the arccosine of x, in radians
Math.asin(x)        // Returns the arcsine of x, in radians
Math.atan(x)        // Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
Math.ceil(x)        // Returns the value of x rounded up to its nearest integer
Math.cos(x)         // Returns the cosine of x (x is in radians)
Math.exp(x)         // Returns the value of Ex
Math.floor(x)       // Returns the value of x rounded down to its nearest integer
Math.log(x)         // Returns the natural logarithm (base E) of x
Math.max(x, y, z, ..., n)  // Returns the number with the highest value
Math.min(x, y, z, ..., n)  // Returns the number with the lowest value
Math.pow(x, y)      // Returns the value of x to the power of y
Math.random()       // Returns a random number between 0 and 1
Math.round(x)       // Returns the value of x rounded to its nearest integer
Math.sin(x)         // Returns the sine of x (x is in radians)
Math.sqrt(x)        // Returns the square root of x
Math.tan(x)         // Returns the tangent of x
```

[1.3.3] Boolesche Variable

Variable können durch **Vergleichsoperatoren** miteinander verglichen werden.

Operator	Beschreibung	Beispiel (x = 6; y = 5)
==	gleich wie	x == 5; // false
!=	ungleich wie	x != 5; // true

===	gleicher Wert und gleicher Typ	x === y; // false
		x === 6; // true

!==	unterschiedlicher Wert oder unterschiedlicher Typ	x !== y; // true
		x !==6; // false

>	größer als	x > y; // true
<	kleiner als	x < y; // false

>=	größer als oder gleich wie	x >= y; // true
		x >= 6; // true

<=	kleiner als oder gleich wie	x <= y; // false
		x <= 6; // true

Das Ergebnis eines Vergleichs ist eine Variable die entweder wahr (true) oder falsch (false) ist.
Solche Variable heißen Boolesch, z.B.: a = (5 == 6); (a ist hier false;), b = (5 < 6); (b ist hier true;).

Alle booleschen Variablen können durch **logische Operatoren** miteinander verknüpft werden.
Dazu zählt man NICHT (not, !), UND (and, &&), ODER (or, ||).

Operator	Bedeutung

!	Mit dem logischen **NICHT-Operator** (!) negieren Sie einen Ausdruck. Sie können also aus »wahr« »falsch« machen und umgekehrt.
	`if( !(ival > 0) ) {`
	`  alert("ival ist nicht größer als 0");`
	`}`

Operator	**Bedeutung**

&&	Ausdrücke, die mit dem **UND-Operator** verknüpft sind, geben nur dann wahr (true) zurück, wenn alle Ausdrücke wahr (true) sind. Zum Beispiel:
	`if ( ival1 > 0  &&  ival2 > 0 ) {`
	`  // Beide Ausdrücke sind wahr = true.`
	`}`

||	Ausdrücke, die mit dem logischen **ODER-Operator** verknüpft sind, geben wahr (true) zurück, wenn mindestens einer der Ausdrücke wahr ist.
	`if ( ival1 > 0  ||  ival2 > 0 ) {`
	`  // Mindestens ein Ausdruck ist wahr = true.`
	`}`

Beispiele:
((x > 0) && (x < 10)) liefert Zahlen zwischen 0 **und** 10.
((x % 2) == 0) liefert nur gerade Zahlen.
((x % 2) != 0) liefert nur ungerade Zahlen.
(((x % 2) == 0) || ((x % 3) == 0)) liefert Zahlen, die durch 2 **oder** durch 3 teilbar sind.

Anmerkung: Mithilfe von ***typeof(Variable)*** kann der aktuelle Datentyp einer Variablen ermittelt werden.
Als Ergebnis werden dann *undefined, string, number, boolean, object* oder *function* zurückgeliefert.

[1.4] Die Kontrollstrukturen

In diesem Kapitel sollen die Kontrollstrukturen für den Programmablauf besprochen werden.

- Einfache Entscheidungen (***if***)
- Zweifache Entscheidungen, (***if - else***)
- Mehrfache Entscheidungen (***switch***)
- Unbedingte Zählschleifen (***for***)
- Bedingte Wiederholungsschleifen (***while***)

[1.4.1] Einfache Entscheidungen (*if*)

Mit bedingten Anweisungen wird der Ablauf des Programmes abhängig von einer Bedingung in verschiedener Weise gesteuert. Bei der einfachen ***if***-Entscheidung wird zu einem bestimmten **Anweisungsblock** verzweigt. Ein Anweisungsblock besteht aus mehreren Anweisungen, die zwischen einer öffnenden geschweiften Klammer ({) und einer schließenden geschweiften Klammer (}) stehen. Ein solcher Anweisungsblock wird durch KEIN Semikolon abgeschlossen. Die Bedingung besteht immer in der Überprüfung eines booleschen Wahrheitswertes (true, false) . Es folgen einige Beispiele zur Demonstration.

var x = 5; y = 6; z = ' Herbert'; // drei nebeneinander definierte Variable

if (x == y) { alert(' Dieser Vergleich ist falsch ! '); }
if (x > y) { alert(' Dieser Vergleich ist falsch ! '); }
if (x != y) { alert(' Dieser Vergleich ist richtig ! '); }
if (x < y) { alert(' Dieser Vergleich ist richtig ! '); }
if (isNaN(x)) { alert(' Dieser Vergleich ist falsch ! '); }
if (isNaN(z)) { alert(' Dieser Vergleich ist richtig ! '); }

Die vordefinierte Funktion ***isNaN(z)*** prüft, ob die Variable ***z*** **keine** Zahl ist - dann liefert sie den Wahrheitswert ***true***, andernfalls den Wahrheitswert ***false***. „isNaN(z)“ bedeutet: z is NOT a NUMBER.

Die vordefinierte Funktion ***result = confirm(Fragetext)*** zeigt einen Fragetext in einem Fenster an und enthält die Schalter <***OK***> und <***Abbrechen***>. Nach Betätigung wird entweder ***true*** oder ***false*** geliefert.

```
<!doctype html>
<html>
<head>
   <title> JS06 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>
<body>
   <p> Programm JS06: Eingabe, Ausgabe und Abfragen von Daten (1). </p>
   <script>
      var eingabe = '';
      eingabe = prompt('Ihren Namen (oder nichts) eingeben',eingabe);
      eingabe = eingabe.trim();
      if (eingabe == '') {
         result = confirm('Wollen Sie namenlos bleiben?')
         if (!result) {eingabe = prompt('Bitte Ihren Namen nochmals eingeben',eingabe);}
      }
      eingabe = prompt(eingabe + ': Bitte Ihr Alter eingeben?');
      eingabe = eingabe.trim();
      if (eingabe == '')  {alert('Ihre Eingabe ' + eingabe + ' ist leer!');}
      if (isNaN(eingabe)) {alert('Ihre Eingabe ' + eingabe + ' ist keine Zahl!');}
      if (eingabe > 60)   {alert('Mit ' + eingabe + ' sind Sie bereits in Pension!');}
   </script>
</body>
</html>
```

[1.4.2] Zweifache Entscheidungen (*if - else*)

Häufig wird die ***if***-Anweisung ergänzt durch eine ***else***-Anweisung, die nur dann ausgeführt wird, wenn die zuvor überprüfte ***if***-Bedingung ***false*** gewesen ist.

```
<!DOCTYPE html>
<html>
<head>
   <title> JS07 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>
<body>
   <p> Programm JS07: Eingabe, Ausgabe und Abfragen von Daten (2). </p>
   <script>
      const erwachsen = 18;
      var info = 'Bitte Alter ( > ' + erwachsen + ' Jahre) eingeben';
      var name = '???';
      var alter = '';
      name = prompt('Bitte Ihren Namen eingeben', name);
      alter = prompt(info);
      if ((name == '???') || (alter <= 18)) {
         alert('Zutritt verboten. Sie sind namenlos oder zu jung!');
      }
      else {
         var ausgabe = 'Hallo ' + name + '. Bitte eintreten !';
         alert(ausgabe);
      }
   </script>
</body>
</html>
```

[1.4.3] Mehrfache Entscheidungen (*switch*)

Wenn mehrere Fälle überprüft werden sollen, dann können dafür mehrere ***if***-Abfragen verwendet werden. Einfacher ist es jedoch die Fallunterscheidungen mit der ***switch***-Anweisung auszuführen. Im nachfolgenden Programm wird eine Zahl von 1 bis 6 gewürfelt und es soll entschieden werden, welche Zahl gewürfelt worden ist. Die vordefinierte Funktion ***Math.random();*** liefert Zufallszahlen zwischen 0 und 1 (inklusive 0 und exklusive 1). Die vordefinierte Funktion ***Math.floor(x)*** liefert die zu ***x*** nächstkleinere ganze Zahl. ***Math.floor(6*Math.random()) + 1;*** liefert ein Würfelergebnis.

```
<!DOCTYPE html>
<html>
<head>
   <title> JS08 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>
<body>
   <p> Programm JS08: Der Zufall wuerfelt. </p>
   <script>
      var max = 6;
      var wurf = Math.floor( max * Math.random() ) + 1;
      switch(wurf) {
         case 1: { alert('Wurfergebnis = 1'); break; }
         case 2: { alert('Wurfergebnis = 2'); break; }
         case 3: { alert('Wurfergebnis = 3'); break; }
         case 4: { alert('Wurfergebnis = 4'); break; }
         case 5: { alert('Wurfergebnis = 5'); break; }
         case 6: { alert('Wurfergebnis = 6'); break; }
      }
   </script>
</body>
</html>
```

[1.4.4] Unbedingte Zählschleife (*for*)

Soll ein Anweisungsblock (Schleife) mehrmals wiederholt werden, dann kann das mit einer ***for***-Anweisung erfolgen, wenn die Anzahl der Wiederholungen (beispielsweise ***anz***) bekannt ist. Benötigt wird dazu eine Schleifenvariable (beispielsweise ***n***), welche dann innerhalb der Schleife beginnend bei einem Initialwert erhöht (inkrementiert) oder erniedrigt (dekrementiert) wird, so lange eine bestimmte Schleifenbedingung erfüllt ist.

for (Initialwert; Schleifenbedingung; Inkrement) { Anweisungsblock }
for (n = 1; n <= anz; n++) { Anweisungsblock }

```
<!DOCTYPE html>
<html>
<head>
   <title> JS09 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>
<body>
   <p> Programm JS09: Der Mittelwert (Version 1). </p>
   <script>
      var x, n, anz, sum, mwt;
      anz = prompt('Anzahl der Daten (1 < n < 11) eingeben');
      if ( isNaN(anz) ) { alert('Abbruch - Eingabe ist keine  Zahl.'); exit; }
      if ( (anz <= 1) || (anz >= 11) ) {
         alert('Abbruch - ungültige Anzahl !'); exit;
      }
      anz = 1 * anz;
      sum = 0;
      for (n = 1; n <= anz; n++) {
         x = prompt(n + '-te Zahl eingeben (maximal ' + anz + ')');
         if ( isNaN(x) ) { alert('Abbruch - Eingabe ist keine  Zahl.'); exit; }
         sum = sum + parseFloat(x);
      }
      mwt = sum / anz;
      alert('Mittelwert von ' + anz + ' Zahlen = ' + mwt.toFixed(2));
   </script>
</body>
</html>
```

Das nachfolgende Programm demonstriert die Berechnung des Mittelwertes mit Hilfe einer dekrementierenden Schleife: ***for (n = anz; n >= 1; n--) { Anweisungsblock }***

```
<body>
   <p> Programm JS10: Der Mittelwert (Version 2). </p>
   <script>
      var x, n, anz, sum, mwt;
      anz = prompt('Anzahl der Daten (1 < n < 11) eingeben');
      if ( isNaN(anz) ) { alert('Abbruch - Eingabe ist keine  Zahl.'); exit; }
      if ( (anz <= 1) || (anz >= 11) ) {
         alert('Abbruch - ungültige Anzahl !'); exit;
      }
      anz = 1 * anz;
      sum = 0;
      for (n = anz; n >= 1; n--) {
         x = prompt(n + '-te Zahl eingeben (maximal ' + anz + ')');
         if ( isNaN(x) ) { alert('Abbruch - Eingabe ist keine  Zahl.'); exit; }
         sum = sum + parseFloat(x);
      }
      mwt = sum / anz;
      alert('Mittelwert von ' + anz + ' Zahlen = ' + mwt.toFixed(2));
   </script>
</body>
```

Anmerkung:
Der Befehl ***break*** bricht eine Schleife ab und setzt nach ihrem Ende das Programm fort.
Der Befehl ***exit*** hingegen bricht ein Programm vollständig ab, weil es ihn gar nicht gibt, und dieser Syntaxfehler dann zu einem sofortigen Abbruch des Programmes führt.

[1.4.5] Bedingte Wiederholungsschleife (*while*)

Die kopfgesteuerte ***while***-Schleife wird so lange ausgeführt, wie die ***while***-Bedingung erfüllt ist.
Der entsprechende Syntax lautet: ***while (Bedingung) { Anweisungsblock }***

```
<!DOCTYPE html>
<html>
<head>
   <title> JS11 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>
<body>
   <p> Programm JS11: Der Mittelwert (Version 3). </p>
   <script>
      var n = 1;
      var sum = 0;
      var mwt = 0;
      var x = '';
      while (x != '#') {
         x = prompt('Zahl ' + n + ' (Ende mit "#")');
         x = x.trim();
         if ( !isNaN(x) && (x != "") ) {
            sum = sum + parseFloat(x);
            n++;
        }
      }
      anz = n - 1;
      if (anz > 0) {
         mwt = sum / anz;
         alert('Mittelwert von ' + anz + ' Zahlen = ' + mwt.toFixed(2));
      }
      else { alert('Abbruch: Ungültige Dateneingabe!'); }
   </script>
</body>
</html>
```

Bei der fußgesteuerten ***do - while -*** Schleife wird die Schleifenbedingung erst am Ende überprüft, wenn der komplette Anweisungsblock ausgeführt worden ist. Der entsprechende Syntax lautet:

do {
Anweisungsblock
} while (Bedingung);

```
<body>
   <p> Programm JS12: Der Mittelwert (Version 4). </p>
   <script>
      var n = 1;
      var sum = 0;
      var mwt = 0;
      var x = '';
      do {
         x = prompt('Zahl ' + n + ' (Ende mit #)');
         x = x.trim();
         if ( !isNaN(x) && (x != "") ) {
            sum = sum + parseFloat(x);
            n++;
         }
      } while (x != '#');

      anz = n - 1;
      if (anz > 0) {
         mwt = sum / anz;
         alert('Mittelwert von ' + anz + ' Zahlen = ' + mwt.toFixed(2));
      }
      else { alert('Abbruch: Ungültige Dateneingabe!'); }
   </script>
</body>
</html>
```

Anmerkung: Eine kurze und kompakte Alternative zur bedingten Anweisung ist der so genannte **ternäre Operator**, der aus drei Operanden besteht: **Bedingung ? Wert1 : Wert2**.

```
Beispiel:  z = (x < y) ? 'Adam' : 'Eva';
```

Wenn die Bedingung (x < y) zutrifft, dann erhält die Variable z den Wert `'Adam'`, andernfalls den Wert `'Eva'`.

```
<script>
   var x = 5;
   var y = 6;
   var z = "Adam";
   while (y != 0) {
     info = ' x = ' + 5 + ', Eingabe von y (0=Ende) ';
     y = prompt(info,y);
     z = (x < y) ? "Adam" : "Eva";
     alert('Ergebnis = ' + z);
   }
</script>
```

Die „**while**“-Schleife wird dann abgebrochen, wenn für **y** Null eingegeben wird..

[1.4.6] Fehlersuche

In JavaScript können verschiedene Arten von Fehlern auftreten: Syntax-Error, Type-Error, Range-Error, usw..

Wenn in einem JavaScript-Programm Fehler auftreten und dann ein Absturz des Programms erfolgt, so gibt es unterschiedliche Methoden den Fehler zu suchen. Eine einfache Methode besteht im Setzen von Haltepunkten mit Hilfe der ***alert***-Anweisung, in der auch die Werte von Variablen angezeigt werden, die für den Programmablauf wichtig sind. Man beginnt dabei am Programmanfang und setzt jenen Programmteil, der dem Haltepunkt nachfolgt zwischen zwei spezielle Kommentarklammern /* und */. Dadurch wird dieser Teil nicht mehr durchgeführt. Wenn nun die ***alert***-Anweisung richtig ausgeführt wird, so weiß man, dass der vorangehende Programmteil keinen Fehler aufweist. Wird die ***alert***-Anweisung nicht angezeigt, so muss der Fehler davor liegen. Durch Verschieben solcher Haltepunkte oder durch ein Setzen mehrerer Haltepunkte kann der fehlerhafte Programmcode schrittweise eingegrenzt werden.

```
<script>
   var name = prompt('Bitte einen Namen eingeben','');
   for (var i = 1; i <= 8; i++) {
       alert('Schleifendurchgang i = ' + i + ' von ' + name);
   }
</script>
```

Eine weit verbreitete Art der Überprüfung und Überwachung eines Programmcodes ist die Verwendung der so genannten **Console.** Dieses Objekt ist in allen Browsern vorhanden und kann über deren Eigenschafts-Menü geöffnet werden.

Mit dem Befehl **console.log(*Variablenliste*)** werden in einer Systemmeldung die jeweiligen Werte der Variablen – und auch mögliche Fehler - angezeigt.

```
<script>
   var name = prompt('Bitte einen Namen eingeben','');
   for (var i = 1; i <= 8; i++) {
       console.log('Schleifendurchgang i = ' + i + ' von ' + name);
   }
</script>
```

Neben der **Console** gibt es in allen Browsern noch einen **Debugger**. Mit diesem Werkzeug können ausgewählte Abschnitte des Programmcodes bequem und genau überwacht werden.

[1.5] Funktionen

[1.5.1] Grundlagen

Funktionen sind eigenständige Unterprogramme bestehend aus einem Anweisungsblock, der mit einem Funktionsnamen und optionalen Parameter (Argumenten) aufgerufen wird. Auch kann eine solche Funktion optional einen Wert an den Aufrufer zurückgeben.

In JavaScript sind Funktionen Objekte, die somit Variablen zugewiesen werden können. Sie können auch als Parameter oder als Rückgabewerte von anderen Funktionen aufgerufen werden. Grundsätzlich gibt es in JavaScript vordefinierte Funktionen (z.B. für die Stringverarbeitung) oder benutzerdefinierte Funktionen.

Jede Funktion wird mit dem Schlüsselwort ***function*** eingeleitet. Dahinter folgen der ***Name*** der Funktion und dann ein ***Klammernpaar*** (…). Innerhalb der Klammern kann optional eine Liste von ***formalen Parametern*** stehen, die durch Beistriche getrennt sind. Der eigentliche Code (***Funktionskörper)*** wird zwischen geschweiften Klammern {…} geschrieben. Optional kann im Funktionskörper auch ein Rückgabewert hinter dem Schlüsselwort ***return*** stehen. Die ***return***-Anweisung innerhalb einer Funktion beendet die Funktion sofort.

Beispiel einer Funktionsdeklaration mit zwei formalen Parametern:

```
function addition( parameter1, parameter2 ) {
   var sum = parameter1 + parameter2;
   return sum;
}
```

Beispiel eines Funktionsaufrufes (Ersetzung der formalen Parameter durch reale Argumente):

```
var summe = addition(200, 300);
```

```
<!DOCTYPE html>
<html>
<head>
   <title> JS14 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>
<body>
   <p> Programm JS14: Funktionen. </p>
   <script>
      var eingabe, ausgabe;

      // Eine Funktion "runden" definieren
      function runden(zahl,dezi) {
         var ergebnis = zahl.toFixed(dezi);
         return ergebnis;
      }

      // Eine Funktion "wurzel" definieren
      function wurzel(zahl) {
         var ergebnis1 = Math.sqrt(zahl);
         var ergebnis2 = runden(ergebnis1,4);
         return ergebnis2;
      }

      eingabe = 16;
      eingabe = prompt('Bitte eine Zahl eingeben',eingabe);

      if ( isNaN(eingabe) ) {
         alert('Der Eingabewert "' + eingabe + '" ist keine Zahl.');
         ausgabe = eingabe;
      }
      else {
         var w = wurzel(eingabe);
         ausgabe = 'Die Wurzel von ' + eingabe + ' ist ' + w + '.';
         alert(ausgabe);
      }

      document.write(ausgabe);
   </script>
</body>
</html>
```

[1.5.2] Der Gültigkeitsbereich von Variablen

Jedes JavaScript-Programm besitzt für seine Variable einen **Gültigkeitsbereich** (Scope). Die deklarierten Variablen im Hauptprogramm sind global gültig, d.h. sie gelten auch in jedem Unterprogramm und können dort auch verändert werden (Achtung!).

Die in einem Unterprogramm deklarierten Variablen sind nur dort gültig und heißen daher lokale Variable. Sollen Variable sogar nur in einem Code-Block gültig sein, dann müssen sie dort mit dem Schlüsselwort ***let*** anstelle von ***var*** definiert werden. Das folgende Programm soll diese Sachverhalte mit vier Variablen demonstrieren:

```
u;                          // eine undeklarierte Variable
var x;                      // globale Variable im Hauptprogramm (…1,4)
var y;                      // lokale Variable im Unterprogramm (…3)
let z;                      // lokale Variable im Unterprogramm und
                            // im Code-Block einer if-Anweisung (…2)
```

```
<html>
<head>
  <title> JS15 </title>
  <meta charset="ISO-8859-1">
  <meta name="author" content="Herbert Paukert">
  <style>
     body {background-color: #E8F0D8;}
     p {color: red; font-size: 125%; }
  </style>

  <script>
    var x = 'HERBERT';

    function test() {
      if (true) {
        x = 'RUDOLF';   // Änderung der globalen Variablen x
        var y = 'ADAM';
        let z = 'EVA';
        alert('2, innerhalb des if-Codes: x = ' + x);
        alert('2, innerhalb des if-Codes: var y = ' + y);
        alert('2, innerhalb des if-Codes: let z = ' + z);
        u = 'ZENZ';    // eine undeklarierte Variable u
        alert('2, undeklariert innerhalb des if-Codes: u = ' + u);
      }
      alert('3, außerhalb des if-Codes: x = ' + x);
      alert('3, außerhalb des if-Codes: y = ' + y);
      try { alert('3, außerhalb des if-Codes: z = ' + z); }
      catch(error) { alert(error.message); }
      alert('3, außerhalb des if-Codes: u = ' + u);
    }

  </script>
</head>

<body>
  <p> Programm JS15: Globale und lokale Variable. </p>
  <script>
    alert('1, Globale Variable (original): x = ' + x);
    test();
    alert('4, Globale Variable (geändert): x = ' + x);
    try { alert('4, außerhalb der Funktion: y = ' + y); }
    catch(error) { alert(error.message); }
    try { alert('4, außerhalb der Funktion: z = ' + z); }
    catch(error) { alert(error.message); }
    alert('4, außerhalb der Funktion: u = ' + u);
    alert('4, ACHTUNG: undeklarierte Variable gelten immer global !!!');
  </script>
</body>
</html>
```

Anmerkung: Durch das Schlüsselwort ***try*** wird der anschließende Anweisungsblock nach Fehlern überprüft. Werden solche entdeckt, dann kann nach dem darauf folgenden Schlüsselwort ***catch*** eine entsprechende Fehlermeldung ausgegeben werden.

Neben den **normalen Funktionen** gibt es noch **weitere Funktionen**: anonyme Funktionen, selbstauslösende Funktionen, Arrow-Funktionen, Callback-Funktionen und die so genannten Event-Handler als besondere Callback-Funktionen. Dabei handelt es sich um fortgeschrittene Programmiertechniken. Sie werden später ausführlich beschrieben.

[1.5.3] GGT und KGV von zwei Zahlen

Im nachfolgenden Programm werden in zwei getrennten Funktionsprogrammen der größte gemeinsame Teiler (**ggt**) und das kleinste gemeinsame Vielfache (**kgv**) von zwei Zahlen a, b ermittelt. Dabei erfolgt im zweiten Unterprogramm ein Aufruf des ersten Unterprogramms. Es gilt immer a * b = ggt(a,b) * kgv(a,b).

Der größte gemeinsame Teiler zweier Zahlen x und y wird entsprechend dem Euklidischen Algorithmus berechnet. Zuerst wird die Zahl x durch die Zahl y dividiert und der Rest r bestimmt (r = x % y). Dann wird der Divisor durch den Rest dividiert und der neue Rest berechnet. Dieses Verfahren wird schrittweise so lange fortgesetzt bis der Rest Null ist. Das muss nach endlich vielen Schritten eintreten, weil Dividend und Divisor immer kleiner werden. Jener letzte Rest, der noch größer als Null war, ist dann der gesuchte GGT, weil er alle vorher erzeugten Dividenden und Divisoren teilt und jeder andere gemeinsame Teiler in ihm enthalten ist.

```
<!DOCTYPE html>
<html>
<head>
   <title> JS15a </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>
<body>
   <p> Programm JS15a: GGT und KGV. </p>
   <script>
      // Unterprogramm "ggt"
      function ggt(x,y) {
          do {
            r = x % y;
            x = y;
            y = r;
         }
         while (r > 0);
         return x;
      }
      // Unterprogramm "kgv"
      function kgv(a,b) {
         x = ggt(a,b);
         y = a * b / x;
         return y;
      }
      // Hauptprogramm
      var eingabe = '90,60';
      var z1, z2, z3, z4;
      var ausgabe1, ausgabe2;
      eingabe = prompt('Zwei ganze Zahlen eingeben',eingabe);
      zerlegung = eingabe.split(',');
      z1 = zerlegung[0];
      z2 = zerlegung[1];
      alert('Erste Zahl = ' + z1 + '\n' + 'Zweite Zahl = ' + z2);
      z3 = ggt(z1,z2);
      z4 = kgv(z1,z2);
      ausgabe1 = ' GGT(' + z1 + ',' + z2 + ') = ' + z3;
      ausgabe2 = ' KGV(' + z1 + ',' + z2 + ') = ' + z4;
      alert(ausgabe1 + '\n' + ausgabe2);
      document.writeln(ausgabe1 + ' und ' + ausgabe2);
   </script>
</body>
</html>
```

Die mächtige String-Funktion ***split(',')*** erzeugt aus den beiden durch den Separator (,) getrennten Teilen des Strings ***eingabe*** = ***'90,60'*** ein entsprechendes Array ***zerlegung***. Damit kann auf die Teilstrings mit ***zerlegung[0]*** und ***zerlegung[1]*** zugegriffen werden.

[1.5.4] Rekursive Funktionen

Eine besondere Klasse von Unterprogrammen sind **Rekursionen**. Sie sind Verfahren zur schrittweisen Entwicklung von Datenstrukturen. Vor dem ersten Entwicklungsschritt ist immer eine Anfangsstruktur (Rekursionsanfang) gegeben. Dann ist eine Rekursionsvorschrift vorhanden, welche erklärt, wie man die n-te Entwicklungsstufe S(n) aus der (n-1)-ten Entwicklungsstufe S(n-1) erzeugt. Außerdem muss auch festgelegt sein, wann die Entwicklung beendet ist (Rekursionsabbruch).

Beispiel 1: Die Summe SUM(10) der ersten zehn Zahlen soll rekursiv gebildet werden.

Rekursionsanfang:	Anfang mit SUM(0) = 0
Rekursionsvorschrift:	SUM(n) = SUM(n-1) + n
Rekursionsabbruch:	Ende bei n = 10

Beispiel 2: Das Produkt FAK(10) der ersten zehn Zahlen soll rekursiv gebildet werden.

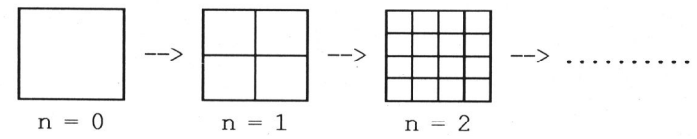
Rekursionsanfang:	Anfang mit FAK(1) = 1
Rekursionsvorschrift:	FAK(n) = FAK(n-1) * n
Rekursionsabbruch:	Ende bei n = 10

Beispiel 3: Die 10-te Potenz POT(x,10) der Zahl x soll rekursiv berechnet werden.

Rekursionsanfang:	Anfang mit POT(x,0) = 1
Rekursionsvorschrift:	POT(x,n) = POT(x,n-1) * x
Rekursionsabbruch:	Ende bei n = 10

Beispiel 4: Ein schachbrettartiger Raster mit 8 x 8 Feldern soll rekursiv gebildet werden.

Rekursionsanfang:	Anfang mit einem Quadrat der festen Seitenlänge a.
Rekursionsvorschrift:	Zerlege das Quadrat in vier kongruente Teilquadrate und führe diese Zerlegung in jedem Teilquadrat durch.
Rekursionsabbruch:	Ende nach 3 Rekursionsstufen (weil $2^3 = 8$).



Eine solche Rekursion wird programmtechnisch dadurch realisiert, dass ein Unterprogramm innerhalb seines eigenen Codes sich selbst aufruft. Dabei werden alle entsprechenden Daten (Parameter, Rücksprungadresse, lokale Variable) auf einen internen Stapelspeicher (Funktions-Stack) abgelegt. Ein Parameter sollte bei jedem Aufruf in gewissem Sinne verändert werden (z.B. immer um Eins verringert). Wird der andauernde Selbstaufruf des Unterprogrammes an eine logische Bedingung in Bezug auf diesen Parameter geknüpft, dann kann ein Abbruch erzwungen werden (z.B. wenn der Parameter Null wird) - andernfalls führen die andauernd aufgestapelten Daten auf dem Stack zu einem Stacküberlauf. Wird eine Abbruchbedingung des rekursiven Selbstaufrufes in der Prozedur gesetzt, dann werden bei der Rückkehr ins Hauptprogramm die gestapelten Daten stufenweise so lange vom Stack abgehoben, bis der allerletzte Rücksprung in das Hauptprogramm zurückführt. Eine rekursive Prozedur enthält somit zwei Ablaufwege, nämlich die Anweisungen VOR und die Anweisungen NACH der Abbruchbedingung (rekursiver Einstieg - rekursive Rückkehr).

In dem nachfolgenden Beispiel wird bei jedem Aufruf der Prozedur ***REK*** der Parameter *n* um Eins verringert. Er wird zur Abbruchbedingung verwendet und steuert somit die Rekursionstiefe. Bei einem Start der Prozedur mit *n = 5* werden folgende Werte für n ausgegeben: *5–4–3–2–1–0–0–1–2–3–4–5*. Die Ausgaben der ersten Werte erfolgen immer beim rekursiven Einstieg, die restlichen bei der Rückkehr.

```
function rek(n) {
   if (n < 0) { return; }
   alert('Inkrement stack :' + n);
   rek(n-1);
   alert('Dekrement stack :' + n);
}
```

```
<!DOCTYPE html>
<html>
<head>
   <title> JS15b, Rekursionen </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>
<body>
   <p> Programm JS15b: Rekursive STACK, SUMME, FAKULTAET, POTENZ und GGT. </p>
   <script>
      // Rekursives Unterprogramm "rek"
      function rek(n) {
         if (n < 0) { return; }
         alert('Inkrement stack :' + n);
         rek(n-1);
         alert('Dekrement stack :' + n);
      }
      // Rekursives Unterprogramm "summe"
      function summe(x) {
         if (x == 0) { return x; }
         else { return (x + summe(x-1)); }
      }
      // Rekursives Unterprogramm "fakult"
      function fakult(x) {
         if (x <= 1) { return x; }
         else { return (x * fakult(x-1)); }
      }
      // Rekursives Unterprogramm "potenz"
      function potenz(x,n) {
         if (n > 0) { return (x * potenz(x,n-1)); }
         else { return 1; }
      }
      // Rekursives Unterprogramm "ggt"
      function ggt(x,y) {
         if  (y == 0) { return x; }
         else { return ggt(y, (x % y)); }
      }
      // Hauptprogramm
      var eingabe1 = 5;
      var eingabe2;
      var sum, fak, z1, z2, z3;
      var ausgabe;
      eingabe1 = 1 * prompt('Eine ganze Zahl eingeben',eingabe1);
      rek(eingabe1);
      eingabe1 = 1 * prompt('Eine ganze Zahl eingeben',eingabe1);
      sum = summe(eingabe1);
      ausgabe = ' Die rekursive Summe von 1 bis ' + eingabe1 + ' ist ' + sum;
      alert(ausgabe);
      fak = fakult(eingabe1);
      ausgabe = ' Die rekursive Fakultaet  ' + eingabe1 + '!  ist  ' + fak;
      alert(ausgabe);
      eingabe2 = '2,4';
      eingabe2 = prompt('Grundzahl und Hochzahl eingeben',eingabe2);
      zerlegung = eingabe2.split(',');
      z1 = 1*zerlegung[0];
      z2 = 1*zerlegung[1];
      z3 = potenz(z1,z2);
      ausgabe = ' Die rekursive Potenz ( ' + z1 + ' hoch ' + z2 + ' ) ist ' + z3;
      alert(ausgabe);
      eingabe2 = '90,60';
      eingabe2 = prompt('Zwei ganze Zahlen eingeben',eingabe2);
      zerlegung = eingabe2.split(',');
      z1 = zerlegung[0];
      z2 = zerlegung[1];
      z3 = ggt(z1,z2);
      ausgabe = ' Der rekursive GGT(' + z1 + ',' + z2 + ') ist ' + z3;
      alert(ausgabe);
   </script>
</body>
</html>
```

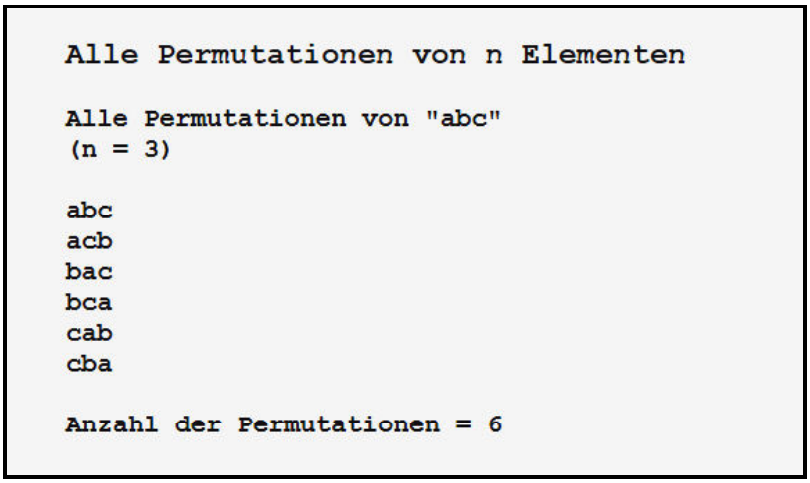
[1.5.5] Backquotes und Templates

```
<!doctype html>
<html>
<head>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: darkred; font-weight: bold;}
   </style>
   <title> JS15d </title>
</head>
<body>
   <h3> Programm JS15d: Backquotes und Templates. </h3>
   <p> Strings können in Javascript von einfachen (') oder doppelten (") Hochkommas<br>
       oder von Backquotes bzw. Backticks (`) eingeschlossen werden. Bei der letzten<br>
       Möglichkeit kann im String auch der $-Operator gesetzt werden, dem dann ein<br>
       von geschweiften Klammern eingeschlossener Ausdruck, eine so genannte Muster-<br>
       vorlage (Template) folgt. Dieses Template kann ein gültiger JavaScript-Ausdruck<br>
       sein, der im String eingebettet ist und dessen Wert dann dort berechnet wird.<br><br>
       In dem vorliegenden Programm werden zwei Strings mittels alert() ausgegeben.<br>
       Im ersten String erfolgt zuerst ein Zeilenvorschub (\n) und dann die Berechnung<br>
       des Quotienten (a / b) der Division von a durch b mithilfe eines Templates.<br>
       Im zweiten String wird der größte gemeinsame Teiler ggt(a,b) rekursiv ermittelt.
   </p>
   <script>
     function ggt(x,y) {
        if (y == 0) { return x; }
        else { return ggt(y,(x % y)); }
     }

     var a = 24, b = 18;
     a = prompt('Eine erste Zahl eingeben',a);
     b = prompt('Eine zweite Zahl eingeben',b);
     alert('Weiter zu den Berechnungen');
     alert(" Erster String: \n Division von " + a + " durch " + b + " = " + `${(a / b)}`);
     alert(" Zweiter String: GGT von " + a + " und " + b + " = " + `${ggt(a,b)}`);
   </script>
</body>
</html>
```

[1.5.6] Komplexe Rekursionen

```
<!doctype html>
<html>
<head>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: darkred; font-weight: bold;}
   </style>
   <title> JS15c </title>
</head>
<body>
   <h3> Programm JS15c: Komplexe Rekursionen. </h3>
   <p>  Eine eingegebene Zahl z soll rekursiv in der Art zerlegt werden,<br>
        dass sie beginnend mit Eins entweder durch Addition mit Drei<br>
        oder durch Multiplikation mit Zwei erreicht wird. Wenn das nicht<br>
        möglich ist, dann wird die Rekursion abgebrochen.
   </p>
   <script>
     function findSolution(target) {
       function find(current, history) {
         if (current == target) { return history; }
         else if (current > target) { return null; }
            else {
            return find(current + 3,`(${history} + 3)`) ||
                   find(current * 2,`(${history} * 2)`);
            }
       }
       return find(1,"1");
     }
     var z = 20;
     z = prompt('Eine ganze Zahl eingeben',z);
     alert(z + " = " + findSolution(z));
   </script>
</body>
</html>
```

[1.5.7] Permutationen



Das Programm "**permut.html**" ermittelt alle Permutationen von n Elementen.

In der Funktion ***allPerms(element)*** wird ein String ***element*** mit ***n*** Elementen als Parameter übergeben. Es sollen nun alle möglichen Anordnungen (Permutationen) der ***n*** Elemente rekursiv erzeugt und auch angezeigt werden. Dazu durchläuft man alle Elemente ***element[i]*** mit i von 1 bis ***n*** und führt dann dabei jedesmal mit den (***n-1***) restlichen Elementen die Funktion ***allPerms(charsRest)*** aus.

Die erzeugten Permutationen werden auf das Array ***results*** geschoben. Bei jedem Aufruf der Rekursion verringert sich die Elementanzahl um Eins. Die Rekursion wird abgebrochen, wenn die Elementanzahl auf Eins schrumpft.

Beispielsweise sollen genau drei Elemente a, b, c permutiert werden: ***abc, acb, bac, bca, cab, cba.***

```
<!DOCTYPE html>
<html>

<head>
<title>Permutationen </title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Permutationen">
<meta name="author" content="Paukert Herbert">
<style>
body { margin:4%; background-color:white; color:black;
       font-family:"Courier New"; font-size:15px; font-weight:bold; }
</style>
</head>

<body>
<h3>Alle Permutationen von n Elementen</h3>

<script>
  function allPerms(element) {
    var results = [];
    if (element.length == 1) {
      count++;
      results.push(element);
      return results;
    }
    for (var i = 0; i < element.length; i++) {
      var charFirst = element [i];
      var charsRest = element.substring(0, i) + element.substring(i + 1);
      var restPerms = allPerms(charsRest);
      for (var j = 0; j < restPerms.length; j++) {
        results.push(charFirst + restPerms[j]);
      }
    }
    return results;
  }
```

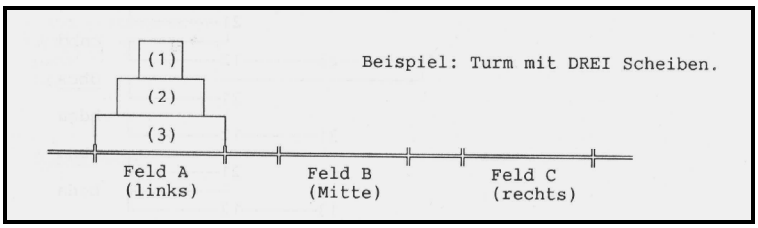
```
  var str0 = "abc";
  var str = prompt('Eingabe der Elemente',str0);
  var anz = str.length;
  var count = 0;

  var erg1 = allPerms(str);     // → Aufruf der Rekursion

  var erg2 = erg1.toString();
  var erg3 = erg2.replace(/,/gi,'<br>');
  document.writeln('Alle Permutationen von "' + str + '"<br>');
  document.writeln('(n = ' + anz + ') <br><br>');
  document.writeln(erg3)
  document.writeln('<br><br>' + 'Anzahl der Permutationen = ' + count + '<br>');
</script>

</body>
</html>
```

[1.5.8] Die Türme von Hanoi

```
Die Türme von Hanoi

In Hanoi gibt es drei Türme (A,B,C). Aufgabe ist es,
alle Scheiben von "A" nach "C" zu transportieren.
Diese sind im Turm "A" der Größe nach geordnet und
sollen auch im Turm "C" der Größe nach geordnet sein.
Die Türme "B" und "C" sind zu Beginn immer leer,
und es gelten dabei folgende Transportregeln:

(T1) Es darf immer nur EINE Scheibe bewegt werden.
(T2) Es muss die Kleinere auf der Größeren liegen.
(T3) Der Turm "B" kann als Zwischenablage dienen.

Lösung bei 3 Scheiben:

1. Lege die oberste Scheibe von A auf C
2. Lege die oberste Scheibe von A auf B
3. Lege die oberste Scheibe von C auf B
4. Lege die oberste Scheibe von A auf C
5. Lege die oberste Scheibe von B auf A
6. Lege die oberste Scheibe von B auf C
7. Lege die oberste Scheibe von A auf C

Anzahl der Transporte = 7
```

Gegeben sind drei Basisfelder für den linken Turm A, den mittleren Turm B und den rechten Turm C, und zusätzlich n verschieden große Scheiben. Diese Scheiben liegen am Anfang auf Feld A nach zunehmender Größe gestapelt. Die Aufgabe besteht darin, den Scheibenturm von Feld A nach Feld C so zu transportieren, dass seine Anordnung erhalten bleibt. Für den Transport gelten drei Regeln:

(T1) Es darf immer nur EINE Scheibe transportiert werden.
(T2) Es darf immer nur eine KLEINERE auf einer GRÖSSEREN Scheibe liegen.
(T3) Das Feld B darf zur Zwischenspeicherung von Scheiben benutzt werden.



Zur Lösung der Aufgabe wird der Turmaufbau rekursiv durchgeführt, indem temporäre Türme aus maximal (n-1) Scheiben gemäß den drei Regeln auf entsprechenden Basisfeldern erzeugt werden. Offenkundig richtet sich der Scheibentransport nach der Anzahl von Turmscheiben auf dem jeweiligen Startfeld. Ist diese ungerade, dann muss die oberste Scheibe direkt auf das jeweilige Zielfeld gelegt werden. Ist die Anzahl hingegen gerade, muss der Scheibentransport auf das jeweilige Zwischenfeld erfolgen. Nach diesen Überlegungen werden nun entsprechende Türme oberhalb der jeweiligen Basis erzeugt – solange bis die Anzahl der transportierten Scheiben auf Eins geschrumpft ist.

```
<!DOCTYPE html>
<html>
<head>
<title>Türme von Hanoi </title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Türme von Hanoi">
<meta name="author" content="Paukert Herbert">
<style>
body { margin:4%; background-color:white; color:black;
       font-family:"Courier New"; font-size:15px; font-weight:bold; }
#aus { color: red; font-style:italic; }
</style>
</head>

<body>
  <h3> Die Türme von Hanoi</h3>
  In Hanoi gibt es drei Türme (A,B,C). Aufgabe ist es,<br>
  alle Scheiben von "A" nach "C" zu transportieren.<br>
  Diese sind im Turm "A" der Größe nach geordnet und<br>
  sollen auch im Turm "C" der Größe nach geordnet sein.<br>
  Die Türme "B" und "C" sind zu Beginn immer leer,<br>
  und es gelten dabei folgende Transportregeln:<br><br>
  (T1) Es darf immer nur EINE Scheibe bewegt werden.<br>
  (T2) Es muss die kleinere auf der größeren liegen.<br>
  (T3) Der Turm "B" kann als Zwischenablage dienen.<br>

  <p id="aus">

  <script>
    function transport(A, B, C, n)
    // Tranportiert n Scheiben von Turm A nach Turm C
    // und verwendet dabei Turm B als Zwischenablage.
    {
      if (n == 1) {
        count++;
        document.writeln(count + ". Lege die oberste Scheibe von " + A + " auf " + C + "<br>");
      }
      else {
        transport(A, C, B, n-1);
        transport(A, B, C, 1);
        transport(B, A, C, n-1);
      }
    }

    var anz0 = 3;
    var anz = prompt('Eingabe der Scheiben-Anzahl',anz0);
    var n = parseInt(anz);
    var count = 0;
    document.writeln('Lösung bei ' + n + ' Scheiben: <br><br>');

    transport("A", "B", "C", n);   // → Aufruf der Rekursion

    document.writeln('<br>' + 'Anzahl der Transporte = ' + count);
  </script>

</p>
</body>
</html>
```

[1.5.9] Restparameter in Funktionen

Wird der Rest-Operator **...** zusammen mit einem Bezeichner (z.B. „numbers“) als letzter Eintrag der Parameterliste einer Funktion notiert, dann erzeugt er beim Funktionsaufruf ein Array mit allen Parametern der Funktion. Im folgenden Beispiel wird die Summe aller Zahlenparameter gebildet.

```
<script>
  alert('Restparameter in Funktionen');
  function addAll(...numbers) {
     let result = 0;
     for (let i = 0; i < numbers.length; i++) {
         result += numbers[i];
     }
     return result;
  }
  alert(addAll(1,2));        // ergibt 3
  alert(addAll(1,2,3));      // ergibt 6
  alert(addAll(1,2,3,4));    // ergibt 10
</script>
```

[1.5.10] Anonyme Funktionen und selbstauslösende Funktionen

Normale Funktionen werden immer **mit** einem Namen deklariert.
Anonyme Funktionen werden jedoch **ohne** Namen deklariert.
Selbstauslösende Funktionen haben keinen Namen und werden am Ort ihrer Deklaration sofort ausgeführt. Sie müssen von einer runden Klammer umfasst werden, und sie schließen danach mit einem Klammernpaar für etwaige Parameter.

```
<!DOCTYPE html">
<html>
<head>
<title> JS16a </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Herbert Paukert">
</head>

<body>
<h3> JS16a: Anonyme und selbstauslösende Funktionen </h3>

<script>
  var z = '2,3';
  z = prompt('Zwei Zahlen eingeben',z);
  var zxy = z.split(',');
  var a = zxy[0];
  var b = zxy[1];

  var sum = function(x,y) { return (1*x + 1*y); }
  alert('Anonyme Funktion:  ' + a + ' + ' + b + ' = ' + sum(a,b));

  (function() {
    var c = 1*a + 1*b;
    alert('Selbstauslösende Funktion:  ' + a + ' + ' + b + ' = ' + c);
  })(a,b);

</script>

</body>
</html>
```

[1.5.11] Arrow-Funktionen

Die so genannten Arrow-Funktionen (=>) sind eine abgekürzte Schreibweise von Funktionen:
(Parameterliste) => **{ Funktionskörper }**, beispielsweise (x,y) => { z = x + y; return 2*z; }
Besteht die Funktion nur aus einer Anweisung, kann die geschweifte Klammer sogar wegfallen.

```
<!DOCTYPE html>
<html>
<head>
<title> JS16b </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Herbert Paukert">

<script>
  function add(x,y) { return (1*x + 1*y); }          // benannte Funktion
  var sum1 = function(x,y) { return (1*x + 1*y); }   // anonyme Funktion
  var sum2 = (x,y) => 1*x + 1*y;                     // Arrow-Funktion 1
  var sum3 = (x,y) => {z = 1*x +1*y; return 2*z; }   // Arrow-Funktion 2
</script>
</head>

<body>
<h3> JS16b: Arrow-Funktionen </h3>

<script>
  var z = '2,3';
  z = prompt('Zwei Zahlen eingeben',z);
  var zxy = z.split(',');
  var a = zxy[0];
  var b = zxy[1];

  alert('Benannte Funktion: x + y   = ' + add(a,b));
  alert('Anonyme Funktion : x + y   = ' + sum1(a,b));
  alert('Arrow-Funktion 1 : x + y   = ' + sum2(a,b));
  alert('Arrow-Funktion 2 : 2*(x+y) = ' + sum3(a,b));
</script>

</body>
</html>
```

[1.5.12] Callback-Funktionen

Bei diesem Funktionstyp ruft eine erste Funktion (aFunc) in ihrem Argument eine zweite Funktion (bFunc) auf, die dann im Ausführungsteil durchgeführt wird: ***function aFunc(bFunc) { bFunc(); }***. Eine Anwendung von „callbacks“ sind die asynchronen Ereignisbehandlungsroutinen (Event-Handler).

```
<!DOCTYPE html">
<html>
<head>
<title> JS16c </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Herbert Paukert">

<script>
  function afunc(bfunc) {      // Erstes Beispiel einer Callback-Funktion
   bfunc();
  }

  function add(x,y,ausgabe) {  // Zweites Beispiel einer Callback-Funktion
    var sum = 1*x + 1*y;
    ausgabe(sum);
  }
</script>
</head>

<body>
<h3> JS16c: Zwei Callback-Funktionen </h3>
<script>
  afunc(function() { alert('Call me back !'); });

  var z = '2,3';
  z = prompt('Zwei Zahlen eingeben',z);
  var zxy = z.split(',');
  var a = zxy[0];
  var b = zxy[1];

  add(a,b,function(sum) { alert('Callback-Summe: ' + sum); });
</script>
</body>
</html>
```

[1.5.13] Event-Handler

Events sind Ereignisse (Mausklick, Tastendruck, …), welche an den HTML-Objekten ausgeführt werden. Mit speziellen Unterprogrammen (Event-Handler) wird auf die Events asynchron zum Hauptprogramm reagiert. Im Programm wird mittels „**addEventListener**“ ein Event-Handler in das HTML-Dokument eingebunden. Dieser führt als callback die Zählfunktion „Counter“ aus, wenn mit der Maus auf den „body“ geklickt wird. Der JavaScript-Zugriff auf „body“ erfolgt mit dem Objekt-Attribut id = „body“ und „document.getElementById(„body“).

```
<!doctype html>
<html>
<head>
   <title> JS16d </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">

   <script>
     document.addEventListener("click", function() {
       document.getElementById("body").onclick = Counter();
     });

     var count = 0;
     function Counter() {
        count++;
        alert('Anzahl der Mausklicks auf das Dokument: ' + count);
     }
   </script>
</head>

<body id = "body">
   <h3>
     JS16d: Callback-Funktionen als Event-Handler<br>
     (mehrmals auf das HTML-Dokument klicken ...)
   </h3>
   <br>
</body>
</html>
```

[1.6] Objekte

[1.6.1] Grundlagen

Ein **Objekt** ist eine Speicherstruktur, die durch bestimmte Eigenschaften (Attribute) und durch bestimmte Funktionen (Methoden) gekennzeichnet ist. In JavaScript gibt es vordefinierte Objekte (z.B. String, Math, Date, …) und benutzerdefinierte Objekte. Die Objekte werden wie Variable mit einem Objekt-Namen (z.B. ***person***) bezeichnet. Will man nun mehrere Objekt-Instanzen (Vertreter) von einem Objektmuster (Basisobjekt, Prototyp) erzeugen, dann ist dafür die Definition einer Konstruktorfunktion sinnvoll.

Im folgenden Programmbeispiel wird ein Basisobjekt ***person*** erzeugt, welches die drei Eigenschaften ***name, groesse*** und ***gewicht*** aufweist. Zusätzlich ist noch die Methode ***info*** definiert, welche das Idealgewicht berechnet und dieses zusammen mit den Eigenschaften in einem Meldungsfenster anzeigt.

Eine zentrale Rolle bei der Konstruktion von Objekten spielt das Schlüsselwort ***this***. Diese Variable ist nichts anderes als ein Verweis auf das jeweilige Objekt. Dadurch wird eine Verbindung zwischen den formalen Eigenschafts-Parametern (***name, groesse, gewicht***) und den inneren Objekt-Eigenschaften ***(x, y, z***) hergestellt. Zusätzlich wird mit ***this*** auch auf die Objekt-Methode ***info*** verwiesen.

Nachdem ein Basisobjekt (Muster) konstruiert worden ist, können davon verschiedene Objekt-Instanzen (z.B. person1, person2, …) mit dem Schlüsselwort ***new*** erzeugt werden. Zum Schluss werden noch die Methoden der beiden Instanzen aufgerufen. Anzumerken ist auch, dass zwischen einem Objekt-Bezeichner und den Bezeichnern für Eigenschaften und Methoden immer ein Punkt (**.**) geschrieben werden muss (Punkt-Operator).

```
<!DOCTYPE html>
<html>
<head>
   <title> JS18a </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>
<body>
   <p> Programm JS18a : Objekte. </p>
   <script>
      // Ein Basisobjekt definieren

      function person(name,groesse,gewicht) {
      // Konstruktor für das Person-Objekt
      // mit drei Attributen und einer Methode
         this.x = name;
         this.y = groesse;
         this.z = gewicht;
         this.info  = function() {
            idealgew = (this.y - 100) * 0.90;
            meldung1 = this.x + ', ' + this.y + ', ' + this.z;
            meldung2 = 'Idealgewicht = ' + idealgew.toFixed(2);
            alert(meldung1 + '\n' + meldung2);
         }
      }
      // Zwei Instanzen des Basisobjektes anlegen

      var person1 = new person();
      person1.x = 'Adam';
      person1.y = 177;
      person1.z = 85;

      var person2 = new person();
      person2.x = 'Eva';
      person2.y = 166;
      person2.z = 55;

      // Die zwei Objektinstanzen ausgeben
      person1.info();
      person2.info();
   </script>
</body>
</html>
```

[1.6.2] Vererbung

Wenn ein Basisobjekt mit einer Konstruktor-Funktion angelegt wurde, dann lassen sich davon Objektinstanzen ableiten, die ihre Eigenschaften und Methoden von dem Basisobjekt (Prototyp) übernehmen. Die **Vererbung** ist eine wesentliche Funktionalität von objektorientierten Sprachen.

Will man nun nachträglich neue Eigenschaften und Funktionen dem Basisobjekt hinzufügen, dann muss dafür der ***Prototyp*** der betreffenden Objekte verwendet werden. In dem folgenden Programm wird damit zum Konstruktor des Objektes „person“ eine neue Eigenschaft und eine neue Methode hinzugefügt. Diese werden dann automatisch auf alle Objektinstanzen vererbt. Mithilfe des Schlüsselwortes „***prototype***“ kann auch eine Eigenschaft aus dem Objektkonstruktor entfernt werden. Zu erwähnen ist noch, dass in JavaScript alle Merkmale eines Objektes nicht versteckt gekapselt (private) , sondern immer öffentlich (public) zugreifbar sind.

```
<!doctype html>
<html>
<head>
   <title> JS18b </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>

<body>
   <p> Programm JS18b: Objekte (2) </p>
   <script>
      // Ein Basisobjekt mittels Konstruktor definieren
      // mit einer Eigenschaft und einer Methode
      function person(name) {
         this.x = name;
         this.info  = function() {
            meldung1 = 'Mein Name ist ' + this.x;
            alert(meldung1);
         }
      }
      // Eine Instanz des Basisobjektes anlegen
      var person1 = new person();
      person1.x = 'Adam';

      // Die Objektinstanz ausgeben
      person1.info();

      // Eine neue Eigenschaft zum Objekt-Konstruktor hinzufügen
      person.prototype.sprache = "Deutsch";

      // Eine neue Methode zum Objekt-Konstruktor hinzufügen
      person.prototype.info1 = function() {
            meldung1 = 'Mein Name ist ' + this.x;
            meldung2 = 'Ich spreche ' + this.sprache;
            alert(meldung1 + '\n' + meldung2);
      }
      // Die Objektinstanz ausgeben
      person1.info1();

      // Eine bestehende Eigenschaft aus dem Objekt-Konstruktor entfernen
      delete person.prototype.sprache;

      // Die Objektinstanz ausgeben
      person1.info1();
     </script>
</body>
</html>
```

Hinweis: Anstelle einer Konstruktor-Funktion kann ein individuelles Objekt auch in der so genannten Literal-Schreibweise mithilfe der typenspezifischen Klammern definiert werden. Im folgenden Beispiel wird anstelle von „this“ der Objektname direkt verwendet.

```
var person1 = {
    name: 'Adam',
    info() { alert('Mein Name ist ' + person1.name); }
}
```

[1.6.3] Punktkoordinaten

Im nachfolgenden Programm wird ein Objekt ***TPoint*** konstruiert, das als Eigenschaften die zwei kartesischen Koordinaten ***(x, y)*** eines Punktes in der Ebene aufweist. Zusätzlich sind zwei Methoden definiert. Mit der ersten Methode ***len*** wird die Entfernung des Punktes vom Koordinatenursprung berechnet. Mit der zweiten Methode ***win*** wird jener Winkel ermittelt, den der Ortsvektor des Punktes mit der positiven x-Achse einschließt. Diese beiden Werte werden auch die Polarkoordinaten des Punktes genannt.

Im Programm werden zwei kartesische Koordinaten ***(x, y)*** eingegeben. Damit wird dann mit ***P = new TPoint;*** eine neue Objekt-Instanz (d.h. ein Punkt P) erzeugt. Dann werden mit den zwei Objekt-Methoden ***P.len*** und ***P.win*** die Polarkoordinaten ***(r, w)*** des Punktes ermittelt. Diese werden zum Schluss angezeigt.

```
<!DOCTYPE html>
<html>
<head>
   <title> JS18c </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>

<body>
   <p> Programm JS18c: Punktkoordinaten. </p>
   <script>

      function TPoint(tx, ty) {
        // Konstruktor für das Punkt-Objekt (zweidimensional)
        this.x = tx;
        this.y = ty;
        this.len = function() {
           var z;
           z = Math.sqrt(this.x*this.x + this.y*this.y);
           return z.toFixed(2);
        }
        this.win = function() {
           var z;
           if ((this.x < 0) && (this.y == 0))  { z = 180; return z;}
           if ((this.x == 0) && (this.y < 0))  { z = 270; return z; }
           if ((this.x == 0) && (this.y == 0)) { z = 0; return z; }
           z = Math.atan(this.y/this.x);
           if (z == Infinity) { z = 90; return z; }
           z = Math.abs(180*z/Math.PI);
           if ((this.x < 0) && (this.y > 0)) { z = 180 - z; }
           if ((this.x < 0) && (this.y < 0)) { z = 180 + z; }
           if ((this.x > 0) && (this.y < 0)) { z = 360 - z; }
           if (z >= 360 ) {z = z - 360; }
           return z.toFixed(2);
        }
      }

      var eingabe = '4,3';
      eingabe = prompt('Punktkoordinaten x,y eingeben',eingabe);
      zerlegung = eingabe.split(',');
      var x = zerlegung[0];
      var y = zerlegung[1];

      var P = new TPoint;
      P.x = x;
      P.y = y;
      var r = P.len();
      var w = P.win();

      ausgabe1 = ' Punkt P(' + P.x + '/' + P.y + ')';
      ausgabe2 = ' Polarkoordinaten: r = ' + r + ', w = ' + w;
      alert(ausgabe1 + '\n' + ausgabe2);

   </script>
</body>
</html>
```

[1.6.4] Das THIS-Problem und Closure-Funktionen als Lösung

Der Bezug auf ein bestimmtes Objekt erfolgt immer über das Schlüsselwort „**this**“. Eine Objektmethode hängt als Funktion an dem Objekt und wird mittels „**object.function()**“ aufgerufen. Der Bezug von „**this**“ kann verloren gehen, wenn die Funktion außerhalb des Objektkontextes ausgeführt wird. Das passiert vor allem in folgenden drei Fällen: (1) Wenn die Funktion als ein **Event-Handler** registriert wird. (2) Bei zeitabhängigen Aufrufen der Funktion mit „**setTimeout**“ oder „**setInterval**“. (3) Bei Verwendung als **Callback**, d.h. wenn die Funktion als Parameter einer anderen Funktion übergeben wird.

Grundsätzlich können Funktionen entweder global sein oder spezifische Objektmethoden. Im ersten Fall ist der Funktionskontext das globale Objekt „**window**“. Im zweiten Fall ist der Funktionskontext das spezifische Objekt. Abhängig davon liefert „**this**“ jeweils eine andere Objektreferenz.

Im Programm „**this1.html**“ wird das Objekt „elObj“ mit einem Merkmal und einer Methode definiert. Weil alle Methoden öffentlich sind, können sie mithilfe eines Buttons aufgerufen werden.

```
<!DOCTYPE html>
<html>
<head>
  <title> this1.html </title>
  <meta charset="ISO-8859-1">
</head>
<body>

<script>
  var elObj = {
      merkmal: "Herbert",
      start: function() {
        alert("Beim Start wird aufgerufen: " + this.merkmal);
      }
  };
</script>

<p> <b>THIS-Zeiger (1)</b> </p>
<button onclick="elObj.start()"> Hier Klicken </button>
</body>
</html>
```

Im Programm „**this2.html**“ wird das Objekt „elObj“ mit einem Merkmal und einer Methode definiert. In der Methode ist zusätzlich die Funktion „delay“ installiert, deren Aufruf nach einer Zeitverzögerung von 1000 Millisekunden erfolgt. In dieser Funktion wird auf das Objektmerkmal mittels „this“ zugegriffen. Das liefert dann das Fehler-Ergebnis „undefined“, weil im Funktionsblock die this-Referenz nicht mehr auf das Objekt verweist, sondern auf „window“ als Vater der Timerfunktion – und das globale Objekt „window“ besitzt das betreffende Merkmal nicht!

```
<!DOCTYPE html>
<html>
<head>
  <title> this2.html </title>
  <meta charset="ISO-8859-1">
</head>
<body>

<script>
  var elObj = {
      merkmal: "Herbert",
      start: function() {
        delay = function() { alert("Verzögert wird aufgerufen: " + this.merkmal); }
        setTimeout(delay, 1000);
      }
  };
</script>

<p> <b>THIS-Zeiger (2)</b> </p>
<button onclick="elObj.start()"> Hier Klicken </button>
</body>
</html>
```

Nun ist es natürlich wünschenswert, dass bei derartigen Programmkonstruktionen dieser Fehler behoben werden kann. Tatsächlich wurde dafür eine Lösung gefunden, nämlich die so genannten **Closure**-Funktionen.

• Closure-Funktionen

Eine **Closure**-Funktion ist eine Funktion, welche in eine äußere Funktion eingebettet ist. Diese innere Funktion hat nun Zugriff auf alle Variablen des Gültigkeitsbereiches der äußeren Funktion. Das ist deswegen möglich, weil die lokalen Variablen der äußeren Funktion nach deren Beendigung vom Funktions-Stack nicht sofort gelöscht werden, sondern für innere Funktionen erhalten bleiben. Das Programm „**closure.html**“ demonstriert diesen Sachverhalt.

```
<!DOCTYPE html>
<html>
<head>
   <title> closure.html </title>
   <meta charset="ISO-8859-1">
   <style>
      body {background-color: #E8F0D8;}
      p {color: darkred; font-weight: bold;}
   </style>
</head>
<body>
   <h3> Closures </h3>
   <p>  Closures sind Funktionen, welche in eine äußere Funktion eingebettet sind.<br>
        Closures haben auf alle Variablen der äußeren Funktion vollen Zugriff,<br>
        auch nach deren Funktionsabschluss!<br>
        Hier wird einer inneren Funktion (Closure "iFunc") eine Variable ("name")<br>
        von einer äußeren Funktion "aFunc" zur Verfügung gestellt und angezeigt.<br>
   </p>
   <script>
     function aFunc(t) {
       var name = t;
       function iFunc() {
         alert('Der Name ist: ' + name);
       }
       return iFunc;
     }
     var s = 'Hallo';
     s = prompt('Einen Namen eingeben',s);
     var myFunc = aFunc(s);
     myFunc();
   </script>
</body>
</html>
```

Das folgende Programm „**this3.html**“ zeigt, wie das THIS-Problem mithilfe von Closures gelöst wird. Zuerst wird in der Objekt-Methode „start“ auf der lokalen Variablen „ausgabe“ der Merkmalsbezug „this“ abgespeichert. In der inneren, verzögerten Funktion „delay“ kann auf Variable „ausgabe“ zugegriffen werden. Diese Closure-Funktion liefert dann das gewünschte Objekt-Merkmal.

```
<!DOCTYPE html>
<html>
<head>
  <title> this3.html </title>
  <meta charset="ISO-8859-1">
</head>
<body>
<script>
  var elObj = {
      merkmal: "Herbert",
      start: function() {
        var ausgabe = this;
        delay = function() { alert("Verzögert wird aufgerufen: " + ausgabe.merkmal); }
        setTimeout(delay, 1000);
      }
  }
</script>
<p> <b>THIS-Zeiger (3)</b> </p>
<button onclick="elObj.start()"> Hier Klicken </button>
</body>
</html>
```

Auf diese Art und Weise können verschiedene Objekt-Methoden verschachtelt werden, welche dann Zugriff auf die äußeren Variablen haben.

[1.6.5] „call“ und „apply“

Die **call**-Methode ist eine Methode, welche auf ein Objekt (z.B. „person“) angewendet werden kann. Der erste Parameter ist immer ein Bezugsobjekt (z.B. „person1“). Die anderen, optionalen Parameter sind zusätzliche Objektmerkmale.

```
<!DOCTYPE html>
<html>
<head>
  <title> call.html </title>
  <meta charset="ISO-8859-1">
</head>
<body>
<h3> Beispiele von call-Methoden </h3>
<script>
  var person1 = {
      firstName: "Albert",
      lastName : "Meier"
  }
  var person2 = {
      firstName: "Eva",
      lastName : "Müller"
  }
  var person = {
     fullName: function(stadt, land) {
        return this.firstName + " " + this.lastName + ", " + stadt + ", " + land;
     }
  }
  var s = person.fullName.call(person1,"Berlin","Deutschland");
  var t = person.fullName.call(person2,"Wien","Österreich");
  alert("Erste Person = " + s + "\n" + "Zweite Person = " + t);
</script>
</body>
</html>
```

Die **apply**-Methode ist eine Methode, welche auf ein Objekt (z.B. „person“) angewendet werden kann. Der Unterschied zur **call**-Methode ist, dass die optionalen Parameter als Array-Elemente definiert werden.

```
<!DOCTYPE html>
<html>
<head>
  <title> apply.html </title>
  <meta charset="ISO-8859-1">
</head>
<body>
<h3> Beispiele von apply-Funktionen </h3>
<script>
  var person1 = {
      firstName: "Albert",
      lastName : "Meier"
  }
  alert("Erste Person:\n" + person1.firstName + " " + person1.lastName);
  var eingabe = "Eva,Müller";
  eingabe = prompt("Zweite Person:\nVorname,Familienname",eingabe);
  var zerlegung = eingabe.split(",");
  var vn = zerlegung[0];
  var fn = zerlegung[1];
  var person2 = {
      firstName: vn,
      lastName : fn
  }
  var person = {
     fullName: function(stadt, land) {
         return this.firstName + "  " + this.lastName + ", " + stadt + ", " + land;
     }
  }
  var array1 = ["Berlin","Deutschland"];
  var array2 = ["Wien","Österreich"];
  var s = person.fullName.apply(person1,array1);
  var t = person.fullName.apply(person2,array2);
  alert("Apply-Ergebnis:\n" + "Erste Person = " + s + "\nZweite Person = " + t);
</script>
</body>
</html>
```

[1.6.6] Die Verkettung von Funktioen

Zu allen Funktionen existiert das array-ähnliche Objekt „***arguments***“, welches die Funktionsparameter enthält, auf die mit einem Index zugeriffen werden kann. In dem nachfolgenden Programm werden der Funktion „***addAll()***“ beliebig viele Zahlen-Parameter übergeben, welche dann addiert werden.

In JavaScript besteht auch die Möglichkeit zwei Funktionen ***g(f(x))*** zu verketten, wobei immer zuerst die innere Funktion und dann die äußere Funktion ausgeführt wird. Die verkettete Funktion wird im Programm auf die Variable „***chain2***“ gelegt.

Zusätzlich werden die Parameter der einzelnen Funktionen mit einem „***prompt***“-Befehl eingegeben. Die mächtige String-Funktion ***split(',')*** erzeugt aus den drei durch den Separator (,) getrennten Teilen des Strings ***eingabe*** = ***'4,1,5'*** ein entsprechendes Array ***zerlegung***. Auf die Teilstrings wird dann mit ***zerlegung[0]***, ***zerlegung[1]*** und ***zerlegung[2]*** zugegriffen.

Eine direkte Verkettung von mehr als zwei Funktionen ist nicht möglich. Mit der Funktion „***chainAll()***“ können auch drei (oder mehr) Funktionen ***h(g(f(x)))…*** verkettet werden. Dabei wird der „***apply***“-Befehl verwendet, der in einer „***return***“-Funktion aufgerufen wird. Beginnend mit der innersten Funktion werden die einzelnen Funktionswerte hintereinander berechnet – solange bis die äußerste Funktion erreicht ist.

```
<!DOCTYPE html>
<html>
<head>
<title>chain.html</title>
<meta charset="ISO-8859-1">
</head>
<body>
<h3>Funktionen und ihre Verkettung</h3>
<script>

   function addAll() {
     let result = 0;
     for (let i=0; i < arguments.length; i++) {
        result += arguments[i];
     }
     return result;
   }
   var add = addAll(1,2,3,4);
   alert("Addition (1+2+3+4) = " + add);

   var eingabe = "4,1,5";
   eingabe = prompt("X-Wert,Summand,Faktor",eingabe);
   var zerlegung = eingabe.split(",");
   var xWert = zerlegung[0];
   var sum = zerlegung[1];
   var fak = zerlegung[2];
   alert(" Drei Funktionen \n f(x)=x+sum \n g(x)=fak*x \n h(x)=Math.sqrt(x) ");

   function f(x)  { return 1*x+1*sum; }
   function g(x)  { return 1*fak*x; }
   function gf(x) { return g(f(x)); }
   var chain2 = gf(xWert);
   alert("Zweifache Funktionsverkettung g(f(x)) = " + chain2 );

   function h(x) { return Math.sqrt(x); }

   function chainAll() {
     var funcs = arguments;
     return function() {
        let args = arguments;
        for (let i = funcs.length-1; i >= 0; i--) {
           args = [funcs[i].apply(this, args)];
        }
        return args[0];
     }
   }
   var chain3 = chainAll(h,g,f);
   alert("Dreifache Funktionsverkettung h(g(f(x))) = " + chain3(xWert) );

</script>
</body>
</html>
```

Hinweis: In JavaScript sind Funktionen und Arrays als Objekte mit Merkmalen und Methoden angelegt !

[1.6.7] Prototypische Vererbung

Grundsätzlich gibt es zwei Arten von Vererbung: die klassenbasierte und die prototypische. Bei der klassenbasierten Vererbung werden am Anfang Objektklassen definiert, welche dann unveränderlich und statisch weiterbestehen. Die prototypische Vererbung hingegen beginnt bei einem Stammvater (***Basisobjekt, Prototyp***). Diesem sind Merkmale und Methoden zugeschrieben, welche dann an seine Nachfahren (Erben) übertragen werden. Dieser Prozess ist jedoch flexibel und dynamisch, d.h. die Merkmale und Methoden sind öffentlich und können jederzeit verändert werden. Das Objektmerkmal „***prototype***“ ist ein Zeiger auf den Prototyp des Objektes.

Das folgende Programm „JS18d.html“ definiert zuerst den Prototyp „Katze“. Davon werden dann drei verschiedene Instanzen mithilfe von Konstruktoren („new Katze“) erzeugt. Dabei werden die Inhalte der geerbten Merkmale übernommen oder abgeändert. Unverändert bleiben nur die Merkmalsbezeichner.

Zum Schluss wird im Prototyp ein Merkmal und eine Methode überschrieben. Das bewirkt dann auch eine Inventaränderung bei den schon vorher erzeugten Objekten. Ganz zum Schluss wird sogar ein Prototyp-Merkmal gelöscht und die Auswirkungen davon werden angezeigt.

```
<!DOCTYPE html>
<html>
<head>
  <title> JS18d, Prototypische Vererbung </title>
  <meta charset="ISO-8859-1">
</head>

<body>
<h3> Beispiele von prototypischen Vererbungen </h3>
<script>

  function Katze() {}
  Katze.prototype.name  = "Katze";
  Katze.prototype.rasse = "Kurzhaar";
  Katze.prototype.miau  = function() {
    alert( "Miau-Miau" );
  };

  var mieze = new Katze();
  var mauze = new Katze();
  var cheri = new Katze();

  mieze.name = "Mieze";
  alert( mieze.name + " : " + mieze.rasse );
  mieze.miau();

  mauze.name = "Mauze";
  mauze.rasse = "Langhaar";
  alert( mauze.name + " : " + mauze.rasse );
  mauze.miau();

  Katze.prototype.rasse = "Ohne Haare";
  alert( cheri.name + " : " + cheri.rasse );
  Katze.prototype.miau  = function() {
    alert("Wau-Wau")
  };
  cheri.miau();

  alert( mieze.name + " : " + mieze.rasse );
  mieze.miau();

  alert( mauze.name + " : " + mauze.rasse );
  mauze.miau();

  delete Katze.prototype.rasse;
  alert( mieze.name + " : " + mieze.rasse + '\n' + mauze.name + " : " + mauze.rasse + '\n' +
         cheri.name + " : " + cheri.rasse );

</script>
</body>
</html>
```

Grundsätzlich sind alle Merkmale und Methoden eines Objektes öffentlich, d.h. sie können von außen zugegriffen und auch verändert werden. Unter Datenkapselung versteht man das genaue Gegenteil, d.h. das Objektinventar ist nach außen hin geschützt und nicht zugreifbar (privat). Eine Datenkapselung kann in JavaScript nur mit programmtechnischen Tricks erreicht werden.

[1.6.8] Die Verwendung von „promise“-Objekten

Normalerweise ist der Programmablauf ***synchron***, d.h. jeder Befehl wird erst dann ausgeführt, wenn die Ausführung des vorangehenden Befehls beendet ist. Soll aber eine Funktion unabhängig vom synchronen Ablauf (***asynchron***) ausgeführt werden, so können vor ihre Funktionsanweisung das Schlüsselwort „**async**“ gestellt und zusätzlich noch ein „**promise**“ -Objekt definiert werden. Ein so genanntes promise-Objekt ist ein Objekt, das den möglichen Erfolg oder Misserfolg einer asynchronen Operation repräsentiert – im Fall des Erfolges mit der callback-Methode „**resolve**“, andernfalls bei Misserfolg mit der callback-Methode „**reject**“.

Auf den return-Wert solcher asynchronen Funktionen kann mit **.then** zugegriffen werden.
Der Aufruf des promise-Objektes in einer asynchronen Funktion kann mit „**await**“ erfolgen.

```
<script>
// erstes Beispiel ***********************************
var keepProm = true;
var Prom = new Promise(function(resolve, reject) {
    if (keepProm) { resolve = alert('The given promise IS KEPT !'); }
    else { reject = alert('The given promise IS NOT KEPT !'); }
});
Prom.resolve;  // liefert hier einen Wert
Prom.reject;   // liefert hier KEINEN Wert

// zweites Beispiel **********************************
async function funA(x) {
  return Promise.resolve(Math.pow(x,2));
}
funA(4).then(alert);

// drittes Beispiel **********************************
function sqr(x) {
   var Prom = new Promise(resolve => resolve(Math.pow(x,2)));
   return Prom;
}
async function funB(x) {
  alert(await sqr(x));
}
funB(5);
</script>
```

Im Programm „**sleep.html**“ wird der Ablauf der asynchronen Funktion ***run(n,t)*** für ***msec*** Millisekunden unterbrochen. Das wird durch das Promise-Objekt in der Funktion ***sleep(msec)*** gesteuert.

```
<!DOCTYPE html>
<html>
<head>
<title> Sleep.html</title>
<meta charset="ISO-8859-1">
</head>

<body>
<h2> sleep.html </h2>
<script>
  function sleep(msec) {
  // Wartefunktion in Millisekunden
     return new Promise(resolve => setTimeout(resolve,msec));
  }

  async function run(n,t) {
  // iteratives (und asynchrones) Unterprogramm
    for (var x = 0; x <= n; x++) {
      await sleep(t);
      var y = Math.pow(x,2);
      var z = 'Das Quadrat von ' + x + ' ist ' + y + '<br>';
      document.write(z);
    }
  }

  inp = '10,1000'
  input =  prompt('Anzahl, Wartezeit eingeben', inp);
  arr = input.split(',');
  anz  = 1 * arr[0];
  zeit = 1 * arr[1];
  run(anz,zeit);
</script>
</body>
</html>
```

[1.7] Arrays

[1.7.1] Grundlagen

Ein **Array** ist ein Speicherbereich, in dem mehrere (meistens gleichartige) Objekte als nummerierte (indizierte) Elemente unter einem Array-Namen (z.B. ***nam***) abgespeichert sind. Auf die einzelnen Array-Elemente wird mithilfe ihrer Nummer (Index) zugegriffen (z.B. ***nam[i]***). Dabei beginnt die Nummerierung im Array immer mit **0**.

Ein solches Array wird mit leeren eckigen Klammern definiert (z.B. ***var nam = [ ];***). Alternativ dazu ist auch folgende Definition möglich: ***var nam = new Array( );***
JavaScript stellt mehrere mächtige vordefinierte Funktionen für Arrays zur Verfügung, beispielsweise:

max = nam.length; // liefert die Anzahl der Array-Elemente
nam.sort(); // sortiert das Array
gefunden = nam.includes(gesucht); // prüft, ob das Element ***gesucht*** im Array vorkommt
position = nam.indexOf(gesucht); // ermittelt den Index vom Element ***gesucht*** im Array

```
<!doctype html>
<html>
<head>
   <title> JS19a </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>
<body>
   <p> Programm JS19a: Arrays (1) </p>
   <script>

      var gesucht;
      var gefunden;
      var position;
      var meldung;

      // Ein leeres Array definieren
      var nam = [];
      var max = 0;

      // Namen in das Array schreiben
      nam[0] = 'Hans'; nam[1] = 'Zenz'; nam[2] = 'Herbert'; nam[3] = 'Eva'; nam[4] = 'Adam';
      max = nam.length;
      alert('Anzahl der Array-Element: ' + nam.length);

      // Ausgabestring erzeugen
      var aus = '';
      for (i = 0; i < max; i++) { aus = aus + '(' + i + ')  ' + nam[i] + '\n'; }
      alert (aus);

      // Array sortieren und ausgeben
      nam.sort();
      aus = '';
      for (i = 0; i < max; i++) { aus = aus + '(' + i + ')  ' + nam[i] + '\n'; }
      alert(aus);

      // Einen Namen suchen
      gesucht = nam[4];
      gesucht = prompt('Gesuchten Namen eingeben',gesucht);
      gefunden = nam.includes(gesucht);
      if ( gefunden ) { meldung = ' "' + gesucht + '"  wurde gefunden.'; }
      else { meldung = ' "' + gesucht + '"  wurde NICHT gefunden.'; }
      alert(meldung);

      position = nam.indexOf(gesucht);
      meldung = ' "' + gesucht + '"  wurde auf Position "' + position + '" gefunden.';
      if ( position > -1 ) { alert(meldung); }

   </script>
</body>
</html>
```

[1.7.2] Verschiedene Sortierverfahren

Die vordefinierte Funktion **arr.sort()** sortiert eine Array „arr“ entsprechend dem Zeichencode. Wenn das Array aber beispielsweise nur ganze Zahlen enthält, dann führt dieses Verfahren zu keinem richtigen Ergebnis. Die Sortierung von beispielsweise [5, 15, 2, 45, 3] ergibt dann [15, 2, 3, 45, 5]. Das richtige Ergebnis [2, 3, 5, 15, 45] erhält man, wenn man die so genannte Compare-Funktion **function(a, b) { return a – b }** als Parameter verwendet:

arr.sort(function(a, b) { return a – b });

Dabei werden schrittweise zwei Werte a und b miteinander verglichen. Wenn (a – b) > 0 ist, dann ist a > b und b wird vor a einsortiert. Wenn (a – b) < 0 ist, dann ist a < b und a wird vor b einsortiert. Wenn (a – b) = 0 ist, dann ist a = b und es erfolgt dann keine Einsortierung. Anmerkung: Wird im Returnwert (b – a) statt (a – b) genommen, so wird absteigend sortiert.

function aMin(arr) { return Math.min.apply(null,arr); } liefert die kleinste Array-Zahl.
function aMax(arr) { return Math.max.apply(null,arr); } liefert die größte Array-Zahl.

```
<!doctype html>
<html>
<head>
   <title> JS19b </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>
<body>
   <p> Programm JS19b: Arrays (2) </p>
   <script>

      // Ein leeres Array definieren
      var nam = [];
      var max = 0;
      var aus = '';

      // Namen in das Array schreiben
      nam[0] = '5'; nam[1] = '15'; nam[2] = '2'; nam[3] = '45'; nam[4] = '3';
      max = nam.length;
      alert('Anzahl der Array-Element: ' + nam.length);

      // Ausgabestring erzeugen
      aus = nam.toString();
      alert('Unsortiert: ' + aus);

      // Array sortieren und ausgeben
      nam.sort();
      aus = nam.toString();
      alert('Normal sortiert: ' + aus);

      // Array numerisch aufsteigend sortieren und ausgeben
      nam.sort( function(a, b) { return a - b; } );
      aus = nam.toString();
      alert('Numerisch aufssteigend sortiert: ' + aus);

      // Array numerisch absteigend sortieren und ausgeben
      nam.sort( function(a, b) { return b - a; } );
      aus = nam.toString();
      alert('Numerisch abssteigend sortiert: ' + aus);

      // Kleinste Zahl ermitteln und ausgeben
      function aMin(arr) { return Math.min.apply(null,arr); }
      aus = aMin(nam);
      alert('Kleinste Zahl: ' + aus);

      // Größte Zahl ermitteln und ausgeben
      function aMax(arr) { return Math.max.apply(null,arr); }
      aus = aMax(nam);
      alert('Größte Zahl: ' + aus);

   </script>
</body>
</html>
```

[1.7.3] Wichtige Array-Methoden

```
toString()  // Returns an array as string (with ',' as separator-character)
join('sep') // Returns an array as string (with 'sep' as separator-string)
includes()  // Check if an array contains the specified element
indexOf()   // Search the array for an element and returns its position
forEach()   // Calls a function for each array element
map()       // Creates a new array with the result of calling a function for each element
filter()    // Creates a new array with elements that pass a test provided in a function
reduce()    // Returns a single value of a function for all array elements
pop()       // Removes the last element of an array, and returns that element
push()      // Adds new elements to the end of an array, and returns the new length
shift()     // Removes the first element of an array, and returns that element
unshift()   // Adds new elements to the beginning of an array, and returns the new length
slice()     // Selects a part of an array, and returns the new array
sort()      // Sorts the elements of an array
reverse()   // Reverses the order of the elements in an array
splice()    // Adds or Removes elements from an array
```

```
<!doctype html>
<html>
<head>
   <title> JS19c </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>
<body>
   <p> Programm JS19c: Arrays (3) </p>
   <script>
      // zwei leere Arrays definieren
      var nam  = [];
      var nam1 = [];
      var txt;

      // Namen in das Array schreiben
      nam[0] = 'Adam'; nam[1] = 'Eva'; nam[2] = 'Hans'; nam[3] = 'Herbert'; nam[4] = 'Zenz';

      // CSV-String erzeugen aus allen Array-Elementen (CSV = comma separated values)
      var aus = nam.toString();
      alert(aus);

      // einen Namen hinzufügen (am Array-Ende)
      nam.push('Alfred');
      alert(nam.toString());

      // letzten Namen entfernen (vom Array-Ende)
      nam.pop();
      alert(nam.toString());

      // zwei Namen ab vierter Position einfügen (d.h. zweiter Parameter = 0)
      nam.splice(4,0,'Karl','Kurt');
      alert(nam.toString());

      // zwei Namen ab vierter Position entfernen  (d.h. zweiter Parameter = 2)
      nam.splice(4,2);
      alert(nam.toString());

      // die ersten drei Namen in ein neues Array speichern  (1. Parameter = Startposition)
      nam1 = nam.slice(0,3);
      alert(nam1.toString());

      // einen String mittels Array verändern
      var s = 'Ding';
      var arr = s.split('');
      arr[1] = 'u';
      var t = arr.toString(),
      t = t.replace(/,/g,'');
      alert(s + '  wurde verändert zu  ' + t);
   </script>
</body>
</html>
```

Hinweis: So genannte „***Associative Arrays***“ sind Objekte, in denen anstelle von numerischen Indizes benutzerdefinierte Schlüssel (keys) verwendet werden, denen ein Wert (value) zugewiesen wird. Z.B.:
let arr = { name: ' Meier', age: 12, city: ' Wien ' };

[1.7.4] „map“ - „filter“ - „forEach“

Die **map()-**Methode führt für jedes Array-Element eine vom Anwender definierte Funktion aus und speichert die dabei veränderten Werte auf ein neues Array. Die Funktion muss die Parameter (value, index, array) haben. Werden nur die Elementwerte (values) manipuliert, können die zwei restlichen Parameter entfallen. Im folgenden Beispiel werden alle Zahlen des Arrays quadriert.

function myFunc(value, index, array) { return value * value; }

var zahlen1 = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];
var zahlen2 = zahlen1.map(myFunc);

Alternativ als Arrow-Funktion: var zahlen2 = zahlen1.map((value) => (value * value));

Bei der **filter()-**Methode werden entsprechend einer Bedingung die Array-Elemente gefiltert.
Die **forEach()-**Methode funktioniert wie die **map()-**Methode, jedoch mit dem Unterschied, dass kein neues Array erzeugt wird, sondern dass das originale Array geändert wird.

```
<!DOCTYPE html>
<html>
<head>
  <title> JS19d </title>
  <meta charset="ISO-8859-1">
  <meta name="author" content="Herbert Paukert">
  <style>
    body { background-color: #E8F0D8; }
    p { color: red; font-size: 125%; }
  </style>
</head>
<body>
  <p> Programm JS19d: Arrays (4): filter - map - forEach</p>
  <script>
     var sum = 0;
     var arr0 = [0,1,2,3,4,5,6,7,8,9];
     alert(arr0);
     var arr1 = arr0.filter( (value,index,arr0) => (value % 2 == 0) );
     alert(arr1);
     var arr2 = arr1.map( (value,index,arr1) => (value * value) );
     alert(arr2);
     arr2.forEach( (value,index,arr2) => { sum += value; } );
     alert("Quadratsumme der geraden Ziffern = " + sum);

     var such = prompt("Array-Element suchen",5);
     var gefunden = arr0.includes(parseInt(such));
     if (gefunden) {
       let position = arr0.indexOf(parseInt(such));
       alert(such + " in [" + arr0 + "] gefunden an Position " + position);
     }
     else { alert(such + " in [" + arr0 + "] NICHT gefunden!"); }
  </script>
</body>
</html>
```

[1.7.5] Zweidimensionale Arrays

Das folgende Programm „js19e.html“ erzeugt ein zweidimensionales Array „mat“.
Das entspricht einer quadratischen Matrix mit maximal 10 Zeilen und 10 Spalten.

var max = 10;
var mat = new Array(max);

for (var i = 0; i < max; i++) { mat[i] = new Array(max); }

Das Array-Element in der i-ten Zeile und in der j-ten Spalte kann dann mit **mat[i][j]** angesprochen werden. Dabei können i und j Werte von 0 bis 10 annehmen.

```
<!DOCTYPE html>
<html>
<head>
  <title> JS19e </title>
  <meta charset="ISO-8859-1">
  <meta name="author" content="Paukert Herbert">
  <meta name="description" content="Zweidimensionale Arrays">
```

```
  <script>
  var max = 10;
  var maxRow = 0;               // Maximale Zeilenanzahl
  var maxCol = 0;               // Maximale Spaltenanzahl
  var row = new Array(max);     // Eingabezeilen
  var mat = new Array(max);     // zweidimensionale Matrix
  for (var i = 0; i < max; i++) { mat[i] = new Array(max); }

  function mclear() {
    for (var i = 0; i < max; i++) {
      for (var j = 0; j < max; j++) {
        mat[i][j] = 0;
      }
    }
    row.length = 0;
  }

  function input() {
    row.length = 0;
    row.push('1,2,3');
    row.push('1,2,3,4');
    row.push('1,2');
    maxRow = 3;
    maxRow = prompt('Zeilenanzahl eingeben', maxRow);
    var info = 'Eingeben der Daten von Zeile ';
    maxCol = 0;
    var aus = '';
    for (var i = 0; i < maxRow; i++) {
      s = info + i;
      row[i] = prompt(s, row[i]);
      arr = row[i].split(',');
      len = arr.length;
      if ( len > maxCol) { maxCol = len; }
      aus = aus + row[i] + '\n';
    }
    inf1 = 'Maximale Spaltenanzahl = ' + maxCol;
    inf2 = 'Maximale Zeilenanzahl = ' + maxRow;
    aus = 'Eingabedaten: ' + '\n\n' + aus + '\n' + inf1 + '\n' + inf2 + '\n' + ' ';
    alert(aus);
  }

  function matrix() {
    var aus = '';
    for (var i = 0; i < maxRow; i++) {
      for (var j = 0; j < maxCol; j++) {
        arr = row[i].split(',');
        el = arr[j];
        if ( isNaN(el) ) { el = 0; }
        mat[i][j] = el;
        aus = aus + el + ', ';
      }
      aus = aus + '\n';
    }
    aus = 'Matrix: ' + '\n\n' + aus + '\n';
    alert(aus);
  }
</script>
</head>

<body>
<h3>Zweidimensionale Arrays (js19e.html)</h3>
<input type="button" name="btn" value="input" onclick="input()">   
<input type="button" name="btn" value="matrix" onclick="matrix()">   
<input type="button" name="btn" value="clear" onclick="mclear()">  
</body>
</html>
```

[1.7.6] Multiplikation von Matrizen

Das Produkt einer (m,k)-Matrix mit einer (k,n)-Matrix ist eine (m,n)-Matrix mit m Zeilen und n Spalten. Beispiel: Eine (1,3)-Matrix mal einer (3,1)-Matrix ergibt das skalare Produkt der beiden Vektoren.

```
<!DOCTYPE html>
<html>
<head>
<title> JS19f </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Multiplikation von Matrizen">
```

```
<script>
  var max  = 10;
  var arr1 = [];  // Zeilen und Spalten von Matrix1
  var arr2 = [];  // Zeilen und Spalten von Matrix2
  var arr3 = [];  // Zeilen und Spalten von Matrix3 (Matrix1 * Matrix2)

  var mat1 = new Array(max);     // zweidimensionale Matrix1
  for (var i = 0; i < max; i++) { mat1[i] = new Array(max); }

  var mat2 = new Array(max);     // zweidimensionale Matrix2
  for (var i = 0; i < max; i++) { mat2[i] = new Array(max); }

  var mat3 = new Array(max);     // zweidimensionale Matrix3
  for (var i = 0; i < max; i++) { mat3[i] = new Array(max); }

  function minit() {
    for (var i = 0; i < max;  i++) {
      for (var j = 0; j < max; j++) {
        mat1[i][j] = Math.floor(Math.random() * 10);
        mat2[i][j] = Math.floor(Math.random() * 10);
        mat3[i][j] = 0;
      }
    }
  }

  function mmult() {
    z = arr1[0];
    s = arr2[1];
    t = arr1[1];
    for (var i = 0; i < z; i++) {
      for (var j = 0; j < s; j++) {
        el = 0;
        for (var k = 0; k < t; k++) {
           el = el + 1 * (mat1[i][k] * mat2[k][j]);
        }
        mat3[i][j] = el;
      }
    }
  }

  function input() {
    minit();
    inp1 = '4,3'; inp2 = '3,5';
    input1 =  prompt('Zeilen, Spalten von Matrix1 (<10)', inp1);
    input2 =  prompt('Zeilen, Spalten von Matrix2 (<10)', inp2);
    arr1 = input1.split(',');
    arr2 = input2.split(',');
    z1 = 1 * arr1[0].trim();
    s1 = 1 * arr1[1].trim();
    z2 = 1 * arr2[0].trim();
    s2 = 1 * arr2[1].trim();
    if (s1 != z2) { alert('Spalten von M1 <> Zeilen von M2'); return; }
    arr1[0] = z1; arr1[1] = s1;
    arr2[0] = z2; arr2[1] = s2;
    arr3[0] = z1; arr3[1] = s2;
    matrix(mat1,1);
    matrix(mat2,2);
    mmult();
    matrix(mat3,3);
  }

  function matrix(mat,num) {
    document.writeln('<br>  Matrix ' + num + ': <br>');
    if (num == 1) { z = arr1[0]; s = arr1[1]; }
    if (num == 2) { z = arr2[0]; s = arr2[1]; }
    if (num == 3) { z = arr3[0]; s = arr3[1]; }
    for (var i = 0; i < z; i++) {
      zeile = '     ';
      for (var j = 0; j < s; j++) {
        el = mat[i][j];
        zeile = zeile + el + ', ';
      }
      document.writeln(zeile + '<br>');
    }
  }
</script>
</head>

<body>
<h3> Multiplikation von Matrizen (JS19f) </h3>
  <input type="button" name="btn" value="input" onclick="input()">
</body>
</html>
```

[1.7.7] Formatierte Ausgabe von Zufallszahlen

```
<!doctype html>
<html>
<head>
   <title> JS20 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8; font-family:Courier New; font-size: 16px; }
      p {color: darkred; font-weight: bold;}
   </style>
</head>
<body>
   <h3> Programm JS20: Formatierte Darstellung von Zufallszahlen. </h3>
   <p>  Das Programm erzeugt "anz" Zufallszahlen zwischen "min" und "max"<br>
        (inklusive dieser beiden Grenzen). Die Zahlen werden fortlaufend in<br>
        das Array "num" abgespeichert und dann formatiert ausgegeben, d.h.<br>
        mit führenden Nullen und mit einem Zeilenvorschub nach zehn Zahlen.<br>
        Zusätzlich werden die Zahlen sortiert und formatiert ausgegeben.
   </p>
   <script>

     var num = [];   // Zahlen-Array
     var anz = 100;  // Zahlen-Anzahl
     var min = 1;    // Zahlen-Untergrenze
     var max = 100;  // Zahlen-Obergrenze
     var zzei = 10;  // Zahlen pro Zeile

     function inPut() {
     // Sichere Eingaben
       anz = prompt('Anzahl (von 10 bis 100)',anz);
       if ( isNaN(anz) || (anz < 10) || (anz > 100) ) { anz = 100; }
       min = prompt('Minimum (von 1 bis 100)',min);
       if ( isNaN(min) || (min < 1) || (min > 100) )  { min = 1; }
       max = prompt('Maximum (von 1 bis 100)',max);
       if ( isNaN(max) || (max < 1) || (max > 100) )  { max = 100; }
       if ( 1*max <= 1*min) { min = 1; max = 100; }
       alert(' anz = ' + anz + '\n min = ' + min + '\n max = ' + max);
     }

     function zufZahl(ug,og,n) {
     // Erzeugung von n Zufallszahlen z mit ug <= z <= og
       let diff = (og - ug);
       for (let i = 0; i < n;  i++) {
         z = 1*ug + Math.floor(Math.random() * (diff + 1));
         num[i] = z;
       }
     }

     function zeroIns(z,width) {
     // führende Nullen den Zahlen z voranstellen
     // mit der wählbaren Zahlenlänge "width"
       let s = String(z);
       while (s.length < width) { s = "0" + s }
       return s;
     }

     function outPut() {
     // formatierte Ausgabe der Zufallszahlen
     // mit einem Zeilenvorschub nach jeweils "zzei" Zahlen
       var s = '';
       for (let n = 0; n < num.length; n++) {
          if (n % zzei == 0) { s = s + "\n"; }
          s = s + zeroIns(num[n],2) + ',   ';
       }
       alert(s);
     }

     inPut();
     zufZahl(min,max,anz);
     outPut();
     num.sort(function(a,b){return a - b});  // Zahlen sortieren
     outPut();

   </script>
</body>
</html>
```

[1.8] Sortieren und Suchen von Zufallszahlen

Das Speicherarray - Eine Spielwiese für Algorithmen

Das Erlernen des Programmierens ist in Informatik als Werkzeug zur Problemlösung wichtig und unverzichtbar. Dabei darf der Unterricht nicht stehenbleiben bei der mehr oder weniger geschickten Verwendung von visuellen Dialogobjekten, sondern soll an einfachen Beispielen algorithmisches Denken vermitteln. Von zentraler Bedeutung sind dabei Sortier- und Such-Verfahren. Indizierte Speicherarrays sind eine sehr geeignete Datenstruktur zur systematischen Entwicklung solcher Algorithmen. Ein Speicherarray ist nichts anderes als ein Kasten mit Schubladen, welche fortlaufend mit eindeutigen Nummern versehen sind. Dadurch wird ein gezielter Zugriff auf die Daten in den Schubladen möglich. Im vorliegenden Artikel werden einfache Sortier- und Suchverfahren besprochen, die sich alle auf Datenbestände im Hauptspeicher beziehen. Als Daten werden ganze Zufallszahlen verwendet, welche in einem Array abgespeichert sind. Beispielhaft sollen folgende neun Algorithmen besprochen werden, welche in den Programmen (*js21.html*, *js22.html*, …, *js28.html* und *js28a.html*) realisiert sind.

Neben dem numerischen Datentyp (Number) ist die Zeichenkette (String) ein sehr wichtiger Datentyp. Diese besteht aus aufeinander folgenden Bytes, welche als Zeichen (Character) interpretiert werden und auf die über einen Index zugegriffen werden kann. Im Grund ist eine Zeichenkette nichts anderes als ein Speicherarray, in dessen Schubladen Zeichen stehen. Solche Zeichenketten sind die zentralen Bestandteile jeder Textverarbeitung.

Anmerkung: JavaScript stellt für Arrays mächtige vordefinierte Funktionen zur Verfügung, beispielsweise die Sortierfunktion ***array.sort()***. Trotzdem sollen hier als Programmiertraining diese Funktionen vom Leser selbst erstellt werden. Der interessierte Leser kann auch in den neun Algorithmen – soweit wie möglich – die entsprechenden ***for***-Schleifen durch jene mächtigen sechs Array-Funktionen (***sort, filter, map, forEach, includes, indexOf***) ersetzen, wie sie im fogenden Listing dargestellt sind. Dieses kurze Listing demonstriert den Entwurf des „funktionalen Programmierens“ in JavaScript.

```
<script>
  var arr0 = [9,7,8,6,5,3,4,0,2,1];                                    // (01)
  alert(arr0);                                                         // (02)
  arr0.sort();                                                         // (03)
  alert(arr0);                                                         // (04)
  var arr1 = arr0.filter((value,index,arr0) => value % 2 === 0);       // (05)
  alert(arr1);                                                         // (06)
  var arr2 = arr1.map((value,index,arr1) => value * value);            // (07)
  alert(arr2);                                                         // (08)
  var sum = 0;                                                         // (09)
  arr2.forEach((value,index,arr2) => { sum += value; });               // (10)
  alert("Quadratsumme der geraden Ziffern = " + sum);                  // (11)
  var such = prompt("Array-Element suchen",5);                         // (12)
  var gefunden = arr0.includes(parseInt(such));                        // (13)
  if (gefunden) {                                                      // (14)
     let position = arr0.indexOf(parseInt(such));                      // (15)
     alert(such + " in [" + arr0 + "] gefunden an Position " + position);  // (16)
  }                                                                    // (17)
  else { alert(such + " in [" + arr0 + "] NICHT gefunden!"); }         // (18)
</script>
```

Zu Beginn ist ein unsortiertes Array ***arro0*** mit den 10 Ziffern gegeben. In Zeile (03) wird das Array sortiert.
In der Zeile (05) werden aus dem Array ***arro0*** alle geraden Ziffern in das neue Array ***arr1*** gefiltert.
In der Zeile (07) werden alle Elemente von ***arr1*** in das neue Array ***arr2*** kopiert und dort quadriert.
In der Zeile (10) werden alle Elemente von ***arr2*** summiert und das Ergebnis auf die Variable ***sum*** transferiert.
In den folgende Zeilen wird ein Element gesucht und – wenn gefunden – seine Position im Array ermittelt.
Hinweis: Das komplette Programm ist unter „**js19d.html**“ (siehe Seite -47-) gespeichert.

[1.8.1] Erzeugung von Zufallszahlen mit Wiederholung (js21.html)

Der Zahlenbereich (array) ***zahl*** soll mit ***anz*** ganzzahligen Zufallszahlen belegt werden. Dabei können sich die Zahlen wiederholen. Der interne Zufallsgenerator *Math.random()* liefert eine zufällige reelle Zahl ***z*** zwischen *0* und *1 (0 ≤ z < 1)*. Die Anweisung *Math.floor(lim*Math.random())+1;* hingegen liefert eine zufällige ganze Zahl ***z*** zwischen ***0*** und ***lim*** , einschließlich ***lim*** *(0 < z ≤ lim)*.

```
<script>

   var anz  = 100;  // maximale Anzahl von Zufallszahlen
   var lim  = 100;  // lim = obere Zahlengrenze, 1 = untere Zahlengrenze
   var max  = 20;   // maximale Zahlen in einer Ausgabezeile
   var zahl = new Array(anz);  // Array der Zufallszahlen

   // Zufallszahlen MIT Wiederholung im Array abspeichern
   function zuf1() {
      for (i = 1; i <= anz i++) {
         zahl[i] = Math.floor( lim * Math.random() ) + 1;
      }
   }

   // Zufallszahlen formatiert in einem String anzeigen
   function show() {
      var aus = '';
      for (i = 1; i <= anz; i++) {
         if ((i % max) == 0) { aus = aus + ('   ' + zahl[i]).slice(-3) + ', ' + '\n'; }
         else { aus = aus + ('   ' + zahl[i]).slice(-3) + ', '; }
      }
      alert (aus);
   }

   // Sichere Dateneingabe
   do {
      z = prompt('Anzahl der Zufallszahlen (10 <= n <= 100)',anz);
   } while ( isNaN(z) || (z < 10) || (z > 100) );
   z = Math.floor(z);

   anz = z;
   lim = z;
   zuf1();
   show();

</script>
```

Die einzelnen Zahlen des Speicherarrays ***zahl*** sollen in den Zeilen eines Textes so ausgegeben werden, dass in jeder Zeile genau ***max*** Zahlen nebeneinander stehen, durch jeweils ein Komma getrennt. Insgesamt werden dabei ***anz*** Zahlen ausgegeben. Zu beachten sind erstens die sichere Dateneingabe mithilfe der ***while***-Anweisung und zweitens die formatierte Datenausgabe mithilfe der ***sclice***-Methode, wobei ***slice(-n)*** aus einem String die ersten **n** Zeichen herausschneidet.

[1.8.2] Erzeugung von Zufallszahlen ohne Wiederholung (js22.html)

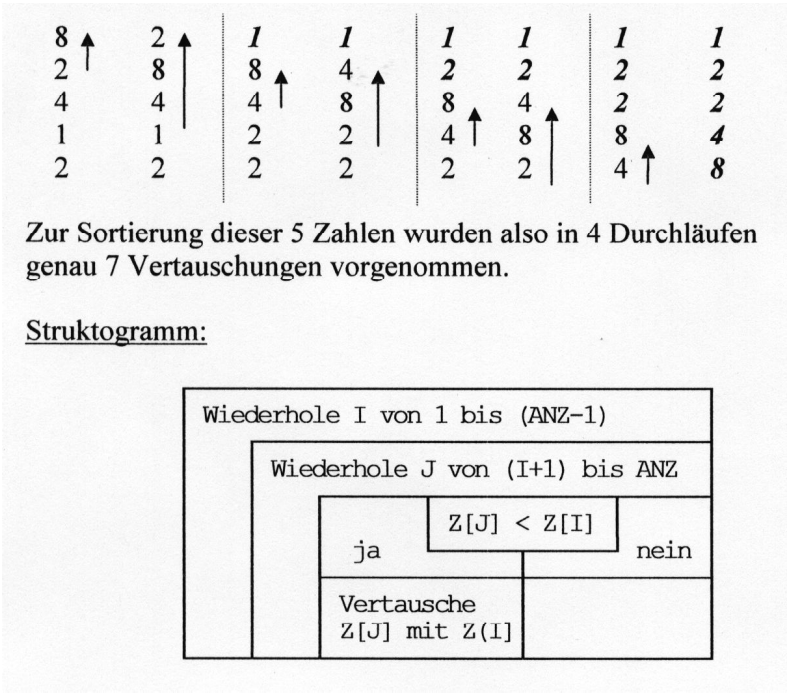
Die Aufgabenstellung ist wie bei [1.7.1]. Es dürfen sich aber jetzt die erzeugten Zufallszahlen **nicht** wiederholen. Am Beginn wird die erste Zufallszahl ***zahl[1]*** erzeugt. Dann erfolgt die schrittweise Bestimmung der restlichen Zahlen. Dabei muss jede neue Zufallszahl mit allen vorher erzeugten Zahlen verglichen werden. Nur wenn sie mit keiner davon übereinstimmt, wird sie in den Bereich (array) übernommen und zur nächsten Zahlen-Erzeugung fortgeschritten.

```
   // Zufallszahlen OHNE Wiederholung im Array abspeichern
   function zuf2() {
      zahl[1] = Math.floor( lim * Math.random() ) + 1;
      for (i = 2; i <= anz i++) {
         do {
           zahl[i] = Math.floor( lim * Math.random() ) + 1;
           gleich = false;
           for (j = 1; j <= (i-1); j++) {
              if (zahl[i] == zahl[j]) { gleich = true; break; }
           }
         } while ( gleich );
      }
   }
```

[1.8.3] Sortieren durch direktes Austauschen (js23.html)

Sortieren und Durchsuchen von Datenbeständen im Hauptspeicher des Computers zählen zu den Grundaufgaben der EDV. Zuerst soll ein einfaches Sortierverfahren (Sortieren durch Austauschen) besprochen werden. Im nächsten Abschnitt wird ein anderer Algorithmus (Sortieren durch Einfügen) erläutert. Neben diesen beiden Verfahren gibt es noch einige andere Techniken, welche wesentlich schneller, aber auch wesentlich komplexer sind (z.B. der rekursive Quicksort-Algorithmus).

In aufsteigender Folge wird jedes Datenelement mit dem ersten verglichen, und wenn es kleiner als dieses ist, dann wird es mit diesem vertauscht. Wenn der ganze Datenbereich durchlaufen ist, steht das kleinste Element an erster Stelle. In einem zweiten Durchlauf wird das zweitkleinste Element an die zweite Stelle befördert, in einem dritten Durchlauf das drittkleinste Element an die dritte Stelle usw. Am Ende aller Durchläufe ist der Bereich sortiert. Zur Illustration sollen fünf Zahlen (8, 2, 4, 1, 2) sortiert werden. Das Array ***zahl*** wird dabei kurz mit ***Z*** bezeichnet.



Zur Sortierung dieser 5 Zahlen wurden also in 4 Durchläufen genau 7 Vertauschungen vorgenommen.

Struktogramm:

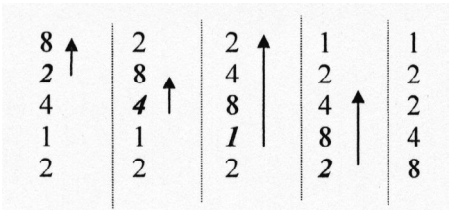

```
<script>

   var anz  = 100;
   var lim  = 100;
   var max = 20;
   var zahl = new Array(anz);

   // Zufallszahlen MIT Wiederholung im Array abspeichern
   function zuf1() {
      for (i = 1; i <= anz i++) {
         zahl[i] = Math.floor( lim * Math.random() ) + 1;
      }
   }

   // Sortieren durch direktes Austauschen
   function sort1() {
      for (i = 1; i <= anz-1; i++) {
         for (j = i+1; j <= anz j++) {
            if (zahl[j] < zahl[i]) {
               x = zahl[i];
               zahl[i] = zahl[j];
               zahl[j] = x;
            }
         }
      }
   }
```

```
   // Zufallszahlen formatiert in einem String anzeigen
   function show() {
      var aus = '';
      for (i = 1; i <= anz; i++) {
         if ((i % max) == 0) { aus = aus + ('   ' + zahl[i]).slice(-3) + ', ' + '\n'; }
         else { aus = aus + ('   ' + zahl[i]).slice(-3) + ', '; }
      }
      alert (aus);
   }

   // Sichere Dateneingabe
   do {
      z = prompt('Anzahl der Zufallszahlen (10 <= n <= 100)',anz);
   } while ( isNaN(z) || (z < 10) || (z > 100) );
   z = Math.floor(z);

   anz = z;
   lim = z;
   zuf1();
   show();
   sort1();
   show();
</script>
```

[1.8.4] Sortieren durch direktes Einfügen (js24.html)

Will man ein neues Datenelement in einen bereits sortierten Bereich richtig einordnen, dann vergleicht man es, beginnend beim letzten Bereichselement, in absteigender Folge mit den einzelnen Elementen des Bereiches. Wenn das neue Element kleiner als ein Bereichselement ist, dann wird das Bereichselement um eine Stelle nach hinten verschoben, wodurch sein alter Platz im Bereich frei wird. Dieses Vergleichen und Verschieben wird so lange fortgesetzt, bis man auf ein Bereichselement trifft, das seinerseits kleiner als das neue Datenelement ist. Dann wird das neue Element an den zuletzt frei gewordenen Platz gestellt und ist somit richtig eingefügt.

Nimmt man nun keine neuen Datenelemente, sondern in aufsteigender Folge die Bereichselemente selbst und fügt sie in die Menge der vorangehenden Bereichselemente richtig ein, dann wird der gesamte Bereich schrittweise sortiert. Je größere Teilbereiche bereits sortiert vorliegen, umso weniger Daten müssen verschoben werden und umso schneller verläuft die Sortierung. Zur Illustration sollen wieder die fünf Zahlen (8, 2, 4, 1, 2) sortiert werden, wobei im Listing Eingabe und Ausgabe der Daten fehlen.

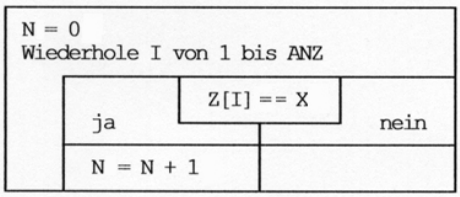



```
   var anz  = 100;
   var lim  = 100;
   var zahl = new Array(anz);

   // Zufallszahlen MIT Wiederholung im Array abspeichern
   function zuf1() {
      for (i = 1; i <= anz i++) {
         zahl[i] = Math.floor( lim * Math.random() ) + 1;
      }
   }

   // Sortieren durch direktes Einfuegen
   function sort2() {
      for (i = 2; i <= anz i++) {
         x = zahl[i];
         j = i - 1;
         while ((x < zahl[j]) && (j > 0)) {
            zahl[j+1] = zahl[j];
            j = j - 1;
         }
         zahl[j+1] = x;
       }
   }
```

[1.8.5] Sequentielles Suchen (js25.html)

Beim sequentiellen Suchen eines gegebenen Datenelementes ***X*** wird der Bereich ***Z***, beginnend am Anfang, schrittweise (***I = 1, ..., ANZ***) durchsucht. Bei Gleichheit von gesuchtem Element ***X*** mit dem jeweilig erreichten Bereichselement ***Z[I]*** wird in einer eigenen Zählvariablen ***N*** mitgezählt. Am Ende des Durchlaufes weiß man dann, wie oft das gesuchte Element im Datenbereich vorkommt. Beim sequentiellen Suchen ist es unwichtig, ob der Bereich bereits sortiert vorliegt oder nicht.

Struktogramm:



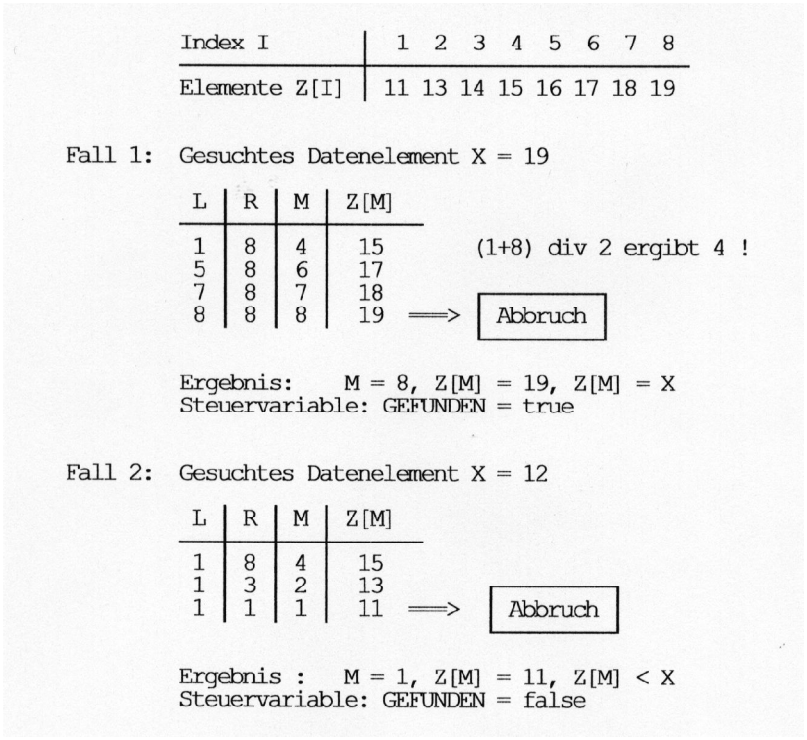
Array Z = zahl
Element X = gesucht

```
<script>

   var anz  = 100;
   var lim  = 100;
   var max = 20;
   var zahl = new Array(anz);
   var gesucht;
   var gefunden;
   var ausgabe;

   // Zufallszahlen MIT Wiederholung im Array abspeichern
   function zuf1() {
      for (i = 1; i <= anz i++) {
         zahl[i] = Math.floor( lim * Math.random() ) + 1;
      }
   }

   // Sequentielles Suchen
   function such1(gesucht) {
      var n = 0;
      for (i = 1; i <= anz i++) {
         if (zahl[i] == gesucht) {n = n + 1; }
      }
      return n;
   }

   // Zufallszahlen formatiert in einem String anzeigen
   function show() {
      var aus = '';
      for (i = 1; i <= anz; i++) {
         if ((i % max) == 0) { aus = aus + ('   ' + zahl[i]).slice(-3) + ', ' + '\n'; }
         else { aus = aus + ('   ' + zahl[i]).slice(-3) + ', '; }
      }
      alert (aus);
   }

   // Sichere Dateneingabe
   do {
      z = prompt('Anzahl der Zufallszahlen (10 <= n <= 100)',anz);
   } while ( isNaN(z) || (z < 10) || (z > 100) );
   z = Math.floor(z);

   anz = z;
   lim = z;
   zuf1();
   show();

   gesucht = zahl[1];
   gesucht = 1 * prompt('Gesuchte Zahl eingeben',gesucht);
   gefunden = such1(gesucht);
   ausgabe = ' Die Zahl  ' + gesucht + '  wurde  ' + gefunden + '-mal  gefunden.';
   alert(ausgabe);
   show();

</script>
```

[1.8.6] Binäres Suchen (js26.html)

Beim binären Suchen muss der Datenbestand bereits sortiert sein. Man zerlegt den Bereich ***Z*** durch Halbierung in zwei Teile und überprüft, ob das gesuchte Datenelement ***X*** im linken oder im rechten Teilbereich liegt. Diesen Teil zerlegt man wieder und wiederholt die fortgesetzte Intervall-Halbierung so lange, bis der in Frage kommende Teilbereich auf ein einziges Element zusammenschrumpft. Entweder ist dieses Element das gesuchte Datenelement, oder der Suchvorgang war nicht erfolgreich. Die Variablen ***L*** und ***R*** bezeichnen die Indizes der Intervallränder und ***M*** ihr arithmetisches Mittel.

Beispiel: Gegeben ist die sortierte Zahlenmenge (11, 13, 14, 15, 16, 17, 18, 19).



Array Z = zahl
Element X = gesucht

```
<script>

   var anz  = 100;
   var lim  = 100;
   var max = 20;
   var zahl = new Array(anz);
   var gesucht;
   var position;
   var ausgabe;

   // Zufallszahlen MIT Wiederholung im Array abspeichern
   function zuf1() {
      for (i = 1; i <= anz i++) { zahl[i] = Math.floor( lim * Math.random() ) + 1;}
   }

   // Binaeres Suchen
   function such2(gesucht) {
      var L = 1;
      var R = anz;
      var gefunden = false;
      while ( (L <= R) && (!gefunden) ) {
         M = Math.round( (1*L + 1*R) / 2 );
         if (gesucht < zahl[M])  { R = M - 1; }
         if (gesucht > zahl[M])  { L = M + 1; }
         if (gesucht == zahl[M]) { gefunden = true; }
      }
      if ( !gefunden )  { M = 0; }
      return M;
    }
```

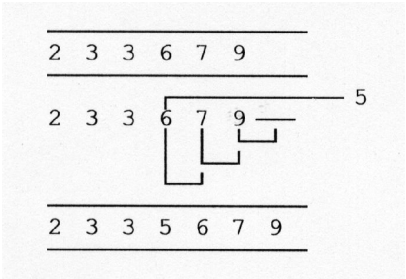
```
      // Sortieren durch direktes Austauschen
      function sort1() {
         for (i = 1; i <= anz-1; i++) {
            for (j = i+1; j <= anz j++) {
               if (zahl[j] < zahl[i]) {
                  x = zahl[i]; zahl[i] = zahl[j]; zahl[j] = x;
               }
            }
         }
       }

      // Zufallszahlen formatiert in einem String anzeigen
      function show() {
         var aus = '';
         for (i = 1; i <= anz; i++) {
            if ((i % max) == 0) { aus = aus + ('   ' + zahl[i]).slice(-3) + ', ' + '\n'; }
            else { aus = aus + ('   ' + zahl[i]).slice(-3) + ', '; }
         }
         alert (aus);
      }

      // Sichere Dateneingabe
      do {
         z = prompt('Anzahl der Zufallszahlen (10 <= n <= 100)',anz);
      } while ( isNaN(z) || (z < 10) || (z > 100) );
      z = Math.floor(z);

      anz = z;
      lim = z;
      zuf1();
      sort1();
      show();

      gesucht = zahl[Math.round(anz/2)];
      gesucht = 1 * prompt('Gesuchte Zahl eingeben',gesucht);
      position = such2(gesucht);
      ausgabe = ' Die Zahl  ' + gesucht + '  wurde an Position  ' + position +
                '  gefunden.';
      alert(ausgabe);
      show();

   </script>
```

Die obige Funktion ***such2*** liefert den Wert **0**, wenn das gesuchte Element ***gesucht*** im Bereich ***zahl*** nicht gefunden wurde. Andernfalls wird die Bereichsposition ***position*** des gefundenen Elementes übergeben.

[1.8.7] Elemente einfügen (js27.html)

Vorausgesetzt wird ein bereits sortierter Datenbereich ***Z.*** Will man ein neues Datenelement ***X*** in diesen Bereich richtig einordnen, dann vergleicht man es beginnend beim letzten Bereichselement in absteigender Folge mit den einzelnen Elementen des Bereiches. Wenn das neue Element kleiner als ein Bereichselement ist, dann wird das Bereichselement um eine Stelle nach hinten verschoben, wodurch sein alter Platz im Bereich frei wird. Dieses Vergleichen und Verschieben wird so lange fortgesetzt, bis man auf ein Bereichselement trifft, das seinerseits kleiner als das neue Datenelement ist. Dann wird das neue Element an den zuletzt frei gewordenen Platz gestellt und ist somit richtig eingefügt. Die unten stehende Funktion liefert zusätzlich noch die Position des eingefügten Elementes. Wichtig ist zu erwähnen, dass nach der Einfügung die Anzahl ***anz*** des Datenbereichs um Eins erhöht wird.

Beispiel: Die Zahl 5 soll in den Bereich (2, 3, 3, 6, 7, 9) eingefügt werden.

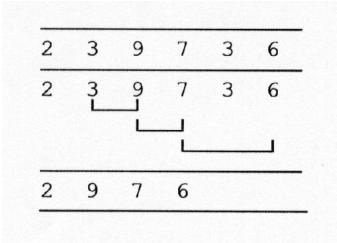


```
<script>
   var anz  = 100;
   var lim  = 100;
   var max  = 20;
   var zahl = new Array(anz);
   var element;
   var position;

   // Array initialisieren
   function init() {
      for (i = 1; i <= 2*anz; i++) { zahl[i] = 0; }
   }

   // Zufallszahlen MIT Wiederholung im Array abspeichern
   function zuf1() {
      for (i = 1; i <= anz; i++) { zahl[i] = Math.floor( lim * Math.random() ) + 1; }
   }

   // Sortieren durch direktes Austauschen
   function sort1() {
      for (i = 1; i <= anz-1; i++) {
         for (j = i+1; j <= anz; j++) {
            if (zahl[j] < zahl[i]) {
               x = zahl[i];  zahl[i] = zahl[j]; zahl[j] = x; }
         }
       }
   }

   // Neues Element einfuegen
   function einfuegen(x) {
      i = anz;
      while ((x < zahl[i]) && (i > 0)) {
         zahl[i+1] = zahl[i];
         i = i - 1;
      }
      zahl[i+1] = x;
      anz = anz + 1;
      return (i+1);
   }

   // Zufallszahlen formatiert in einem String anzeigen
   function show() {
      var aus = '';
      for (i = 1; i <= anz; i++) {
         if ((i % max) == 0) { aus = aus + ('   ' + zahl[i]).slice(-3) + ', ' + '\n'; }
         else { aus = aus + ('   ' + zahl[i]).slice(-3) + ', '; }
      }
      alert (aus);
   }

   // Sichere Dateneingabe
   do {
      z = prompt('Anzahl der Zufallszahlen (10 <= n <= 100)',anz);
   } while ( isNaN(z) || (z < 10) || (z > 100) );
   z = Math.floor(z);
   anz = z;
   lim = z;

   zuf1();
   sort1();
   show();

   element = prompt('Eine Zahl eingeben');
   position = einfuegen(element);
   alert('Die Zahl  ' + element + '  ist an der Position  ' + position + '  eingefuegt.');
   show();
</script>
```

[1.8.8] Elemente entfernen (js28.html)

Das Entfernen eines Datenelementes ***X*** kann in sortierten als auch in nicht sortierten Bereichen ***Z*** erfolgen. Im unten stehenden Verfahren wird der gesamte Bereich durchlaufen und jedes Bereichselement ***Z[I]*** mit dem Datenelement ***X*** verglichen. Nur bei einer Ungleichheit wird das Element im Bereich behalten. Bei Gleicheit wird der ganze Bereich *Z* ohne Datenelement ***X*** neu aufgebaut. Die unten stehende Funktion liefert zusätzlich noch die Anzahl der Element-Entfernungen. Wichtig ist zu erwähnen, dass nach den Entfernungen die Anzahl ***anz*** des Datenbereichs entsprechend verringert wird.

Beispiel: Die Zahl 3 soll aus dem Bereich (2, 3, 3, 6, 7, 9) entfernt werden.




```
<script>
   var anz  = 100;
   var lim  = 100;
   var max  = 20;
   var zahl = new Array(anz);
   var element;
   var gefunden;

   // Zufallszahlen MIT Wiederholung im Array abspeichern
   function zuf1() {
      for (i = 1; i <= anz; i++) { zahl[i] = Math.floor( lim * Math.random() ) + 1; }
   }

   // Zufallszahlen formatiert in einem String anzeigen
   function show() {
      var aus = '';
      for (i = 1; i <= anz; i++) {
         if ((i % max) == 0) { aus = aus + ('   ' + zahl[i]).slice(-3) + ', ' + '\n'; }
         else { aus = aus + ('   ' + zahl[i]).slice(-3) + ', '; }
      }
      alert (aus);
   }

   // Element entfernen
   function entfernen(x) {
      j = 0;
      for (i = 1; i <= anz; i++) {
         if (x != zahl[i]) { j = j + 1;  zahl[j] = zahl[i]; }
      }
      k = anz - j;
      anz = j;
      return k;
   }

   // Sichere Dateneingabe
   do {
      z = prompt('Anzahl der Zufallszahlen (10 <= n <= 100)',anz);
   } while ( isNaN(z) || (z < 10) || (z > 100) );
   z = Math.floor(z);
   anz = z;
   lim = z;
   zuf1();
   show();
   element = zahl[1];
   element = prompt('Eine Zahl eingeben',element);
   gefunden = entfernen(element);
   alert('Die Zahl  ' + element + '  wurde  ' + gefunden + ' -Mal entfernt.');
   show();
</script>
```

[1.8.9] Elemente mischen (shuffle, js28a.html)

Beginnend mit dem letzten Element werden die Elemente des Arrays „*zahl*“ absteigend mit einem zufällig ausgewählten, darunter liegenden Element vertauscht, wobei „*anz = zahl.length*“ ist.

```
function shuffle(zahl,anz) {
   var i,j;
   var count = anz;
   while (count > 1) {
     i = Math.floor(Math.random() * count) + 1;
     j = zahl[count];
     zahl[count] = zahl[i];
     zahl[i] = j;
     count--;
   }
   return zahl;
}
```

[1.9] Dreiecksberechnungen (js29.html)

Von einem Dreieck kennt man die drei Seiten a, b und c. Gesucht sind die Fläche f, der Umfang u, die drei Winkel wA, wB, wC, die drei Höhen ha, hb, hc, der Umkreisradius rU, der Inkreisradius rI. Die gesuchten Werte werden in getrennten Funktionen ermittelt.

```
<!doctype html>
<html>
<head>
   <title> JS29 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>

<body>
   <p> Programm JS29: Das Dreieck. </p>
   <script>
         function rndN(x,n) {
      // Zahl x auf n Dezimalstellen runden
         let z = Math.pow(10,n);
         let y = Math.round(z*x)/z;
         return y;
      }
      function deg(x) {
      // Umwandlung von Bogenmaß (x) in Gradmaß (y)
         let y = 180 * x / Math.PI;
         return y;
      }
      function rad(x) {
      // Umwandlung von Bogenmaß (x) in Gradmaß (y)
         let y = x * Math.PI / 180;
         return y;
      }
      function flaeche(a,b,c) {
      // Fläche eines Dreiecks mit Seiten a, b, c
         let u = (1*a + 1*b + 1*c);
         let s = u / 2;
         let d = s*(s-a)*(s-b)*(s-c);
         if (d <= 0) { return 0; }
         else {
            let f = Math.sqrt(d);
            return rndN(f,3);
         }
      }
      function umfang(a,b,c) {
      // Umfang eines Dreiecks mit Seiten a, b, c
         let f = flaeche(a,b,c);
         if (f == 0) { return 0; }
         else {
           let u = 1*a + 1*b +1*c;
           return rndN(u,3);
         }
      }
      function hoehe(s) {
      // Höhe h auf die Seite s
         let f = flaeche(a,b,c);
         if (f == 0) { return 0; }
         else {
           let h = 2*s/f;
           return rndN(h,3);
         }
      }

      function winkel(a,b,c) {
      // Winkel w zwischen Seite a und Seite b, d.h. gegenüber der Seite c
         let x = (a*a + b*b - c*c) / (2*a*b);
         let y = Math.acos(x);
         let z = deg(y);
         return rndN(z,3);
      }
```

```
      function umkreis(a,b,c) {
      // Umkreisradius r
         let f = flaeche(a,b,c);
         if (f == 0) { return 0; }
         else {
           let r = a*b*c/(4*f);
           return rndN(r,3);
         }
      }
      function inkreis(a,b,c) {
      // Inkreisradius r
         let u = umfang(a,b,c);
         let f = flaeche(a,b,c);
         if (f == 0) { return 0; }
         else {
           let r = 2*f/u;
           return rndN(r,3);
         }
      }
      function seiten(s,x) {
      // Sichere Eingabe der drei Seiten
         let info = 'Dreiecksseiten ' + s;
         let t = prompt(info,x);
         let arr = t.split(',');
         let len = arr.length;
         if (len != 3) { return 0; }
         for (i = 0; i < len; i++) {
            z = 1 * arr[i];
            if ( isNaN(z) || z <= 0 ) { return 0; }
            if (i == 0) { a = z; }
            if (i == 1) { b = z; }
            if (i == 2) { c = z; }
         }
         let e = (a+b+c)/2;
         let d = e*(e-a)*(e-b)*(e-c);
         if (d <= 0) { return 0; }
         return 1;
      }

      // globale Variable
      var abc = '4,3,5';      // Seiten des Dreiecks
      var u, f;               // Umfang, Fläche
      var ha, hb, hc;         // Höhen
      var wA, wB, wC, wSUM;   // Winkel
      var rU, rI;             // Umkreis-, Inkreis-Radius

      var ok = seiten('a,b,c',abc);
      if (ok == 0) {
         aus0 = 'Die Seiten bilden KEIN Dreieck.';
         alert(aus0);
         exit;  // "exit" ist ein "falscher Befehl", der zum Programmabbruch führt
      }

      u = umfang(a,b,c);
      f = flaeche(a,b,c);
      ha = hoehe(a);
      hb = hoehe(b);
      hc = hoehe(c);
      wC = winkel(a,b,c);
      wB = winkel(a,c,b);
      wA = winkel(b,c,a);
      wSUM = wA + wB + wC;
      rU = umkreis(a,b,c);
      rI = inkreis(a,b,c);

      aus1 = 'Dreieck-Seiten: a = ' + a + ', b = ' + b + ', c = ' + c;
      aus2 = 'Umfang U = ' + u + ', Fläche F = ' + f;
      aus3 = 'Höhen: ha = ' + ha + ', hb = ' + hb + ', hc = ' + hc;
      aus4 = 'Winkel: wA = ' + wA + ', wB = ' + wB + ', wC = ' + wC;
      aus5 = 'Winkelsumme = ' + wSUM;
      aus6 = 'Umkreisradius = ' + rU + ', Inkreisradius = ' + rI;

      aus = '\n' + aus1 + '\n\n' + aus2 + '\n' + aus3 +
            '\n' + aus4 + '\n' + aus5 + '\n' + aus6 + '\n\n';
      alert(aus);

   </script>

</body>
</html>
```

[1.10] Datum, Uhrzeit und Zeitmessung

Neben den in Javascript vordefinierten Objekten wie „***string***“ oder „***number***“ gibt es auch ein ***Date***-Objekt, welches alle relevanten Zeitdaten liefert (Tag, Monat, Jahr,Uhrzeit, Zeitzone). Beispielsweise bezieht sich ***jetzt = new Date()*** auf den aktuellen Zeitpunkt. Es stehen mehrere Methoden zum Ermitteln und zum Ändern von Datum und Uhrzeit zur Verfügung. Das folgende Programm demonstriert einige dieser Methoden. Für die Messung der Zeitdauer zwischen zwei Ereignissen kann die Funktion ***getTime()*** verwendet werden. Sie ermittelt, wie viele Millisekunden seit dem 1.Jänner.1970 (0 Uhr) bis zu dem im Date-Objekt bestimmten Zeitpunkt vergangen sind.

```
<!doctype html>
<html>
<head>
   <title> JS30 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body {background-color: #E8F0D8;}
      p {color: red; font-size: 125%; }
   </style>
</head>

<body>
   <p> Programm JS30: Zeitmessung. </p>
   <script>
      var startzeit = 0;
      var endzeit   = 0;
      var zeitdauer = 0;
      var zeit;

      function startClock() { var jetzt = new Date(); startzeit = jetzt.getTime(); }

      function endClock() { var jetzt = new Date(); endzeit = jetzt.getTime(); }

      function calcTime() {
        var dauer = (endzeit - startzeit) / 1000;
        return dauer;
      }

      function fuehrendeNull(zahl) {
        zahl = (zahl < 10 ? '0' : '' )+ zahl;
        return zahl;
      }

      function uhrzeit() {
        var jetzt = new Date(),
        h = jetzt.getHours(),
        m = jetzt.getMinutes(),
        s = jetzt.getSeconds();
        m = fuehrendeNull(m);
        s = fuehrendeNull(s);
        time = h + ':' + m + ':' + s;
        return time;
      }

      alert( Date() );

      zeit = uhrzeit();
      startClock();
      alert(zeit + ' ... Bitte warten Sie einige Sekunden !');
      endClock();
      zeitdauer = calcTime();
      zeit = uhrzeit();
      alert(zeit + ' ... Sie haben ' + zeitdauer + ' Sekunden gewartet !');
   </script>
</body>
</html>
```

[1.11] Reguläre Ausdrücke

Reguläre Ausdrücke sind bestimmte Muster, um Texte zu überprüfen, zu zählen, zu suchen und zu ersetzen. Mit dem so genannten ***RegExp***-Objekt werden reguläre Sprachausdrücke erzeugt, wobei es sich um String-Objekte handelt, welche nach einer speziellen Syntax aufgebaut sind.

(1) Bildung von regulären Ausdrücke mit Literal-Schreibweise: *var reg = /String/;*
(2) Bildung von regulären Ausdrücken mit Konstruktorfunktion: *var reg = new RegExp(String);*
(3) Verwendung von zusätzlichen Steuerzeichen (Flags): *var reg = new RegExp(String, Flags);*
Beispiele für Flags: g = global (d.h. im ganzen String)
i = case-insensitive (d.h. unabhängig von Klein-/Großschrift)
m = multiline (d.h. in allen Textzeilen)

Ein erstes Beispiel: *var str = "In genau diesem Text wird jetzt gesucht.";*

```
(a) var posi = str.search(/Text/);          // Textsuche, ergibt die Position 16
(b) var myArray = str.match(/e/gi);         // Textprüfung, ergibt ein array [e,e,e,e,e,e]
                                            //  g = global, i = case-insensitiv
(c) var count  = str.match(/e/gi).length;   // Textzählung, ergibt die Anzahl 6
(d) var strneu = str.replace(/e/gi, 'XYZ'); // Textersetzung, ersetzt alle „e“ durch „XYZ“
```

Will man anstelle eines konstanten Strings eine String-Variable „such“ in einem regulären Ausdruck verwenden, dann muss sie mit „new RegExp(such, flags)“ registriert werden. Beispielsweise:

```
var str = "In genau diesem Text wird jetzt gesucht.";
var such =  "e“;

var reg = new RegExp(such, 'gi');           // neues RegExp-Objekt mit den Flags 'gi'
                                            // anstelle von var reg = /e/gi;
var count = str.match(reg).length;          // ergibt die Anzahl 6
```

Als zweites Beispiel für die Anwendung von **regulären Ausdrücken** soll die Trimmfunktion dienen. Darunter versteht man eine Funktion, die aus einem String x alle vorangestellten und auch alle nachgestellten Blanks (Leerzeichen) entfernt.

Diese Aufgabe läßt sich zunächst mit der vordefinierten trim-Methode von Strings lösen: **y = x.trim();**

Eine andere Lösung kann mittels regulärer Ausdrücke erfolgen:

```
function  myTrim(x)  {  return x.replace(/^\s+|\s+$/gm,'')  }
```

Erklärung:
- x = String (ev. mit Zeilenvorschüben \n)
- / . . . / = String-Muster mit Spezialzeichen (/ = Begrenzung)
- gm = Steuerzeichen für global (g) und multiline (m)
- ' ' = Nullstring

Bedeutung: Ersetze im String **x** jedes am Zeilenanfang (^) stehende Blank (**\s**) oder (|) jedes am Zeilende (**$**) stehende Blank (**\s**) im ganzen String (**g**) in allen Zeilen (**m**) durch den Nullstring (**' '**).

Das Spezialzeichen + nimmt Bezug auf das davor liegende Zeichen (**\s**) und bewirkt die Gültigkeit für alle diese Zeichen im jeweiligen Kontext, d.h. hier für alle Zeilenanfänge und alle Zeilenenden.

Die vollständige Liste von Spezialzeichen für reguläre Ausdrücke kann in der einschlägigen Fachliteratur nachgelesen werden.

• REPL.html, Suchen und Ersetzen mit regulären Ausdrücken

```
<!DOCTYPE html>
<html>
<head>
   <title> repl.html </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
</head>

<body>
<p>
<b>REPL.HTML, Ersetzen von Stringteilen </b><br>
(1) Zählen eines Zeichens "b" im String s:  count = s.match(/b/gi).length;<br>
(2) Ersetzen von "abc" durch "XYZ" im String s: s = s.replace(/abc/gi,XYZ);
</p>
<script>

  var s = ' abcde abc abc ';
  var t = 'abc';
  var u = 'XYZ'
  s = prompt('Stringeingabe',s);

  var info1 = 'SUCH-String in [' + s + ']';
  t = prompt(info1,t);
  var reg = new RegExp(t,'gi');
  count = s.match(reg).length;
  alert(t + '  kommt in [' + s + '] genau ' + count + '-Mal vor!');

  var info2 = 'ERSATZ-String in [' + s + ']';
  u = prompt(info2,u);
  v = s.replace(reg,u);
  alert(t + '  wurde in [' + s + '] durch ' + u + ' ersetzt:\n [' + v + ']');

</script>
</body>
</html>
```

• TRIM.html, Trimmen mit regulären Ausdrücken

```
<!DOCTYPE html>
<html>
<head>
   <title> trim.html </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
</head>

<body>
<p>
<b>TRIM,HTML, Trimmen eines Strings s </b> (d.h. Entfernen aller randständigen Blanks)<br>
(1) Ohne regulärem Ausdruck mit der Stringfunktion:  s.trim();<br>
(2) Mittels regulärem Ausdruck:  function myTrim(s) { return s.replace(/^\s+|\s+$/gm,'') }<br>
</p>
<script>

  function myTrim(x) { return x.replace(/^\s+|\s+$/gm,''); }

  var s = '          abcde abc abc          ';
  var t = 'abc';
  var u = 'XYZ'

  s = prompt('Stringeingabe mit randständigen Blanks',s);
  var ungetrimmt = s;
  var getrimmt = s.trim();
  var info1 = '1: Trimmen eines Strings ohne regulärem Ausdruck: ';
  alert(info1 + '\n\n ungetrimmt: ' + ungetrimmt + '\n     getrimmt: ' + getrimmt + '\n ');

  var ungetrimmt = s;
  var getrimmt = myTrim(s);
  var info2 = '2: Trimmen eines Strings mittels regulärem Ausdruck: ';
  alert(info2 + '\n\n ungetrimmt: ' + ungetrimmt + '\n     getrimmt: ' + getrimmt + '\n ');

</script>
</body>
</html>
```

Programmieren mit JavaScript, Teil 2
Der Zugriff auf HTML-Objekte

[2.1] Grundlegende Konzepte

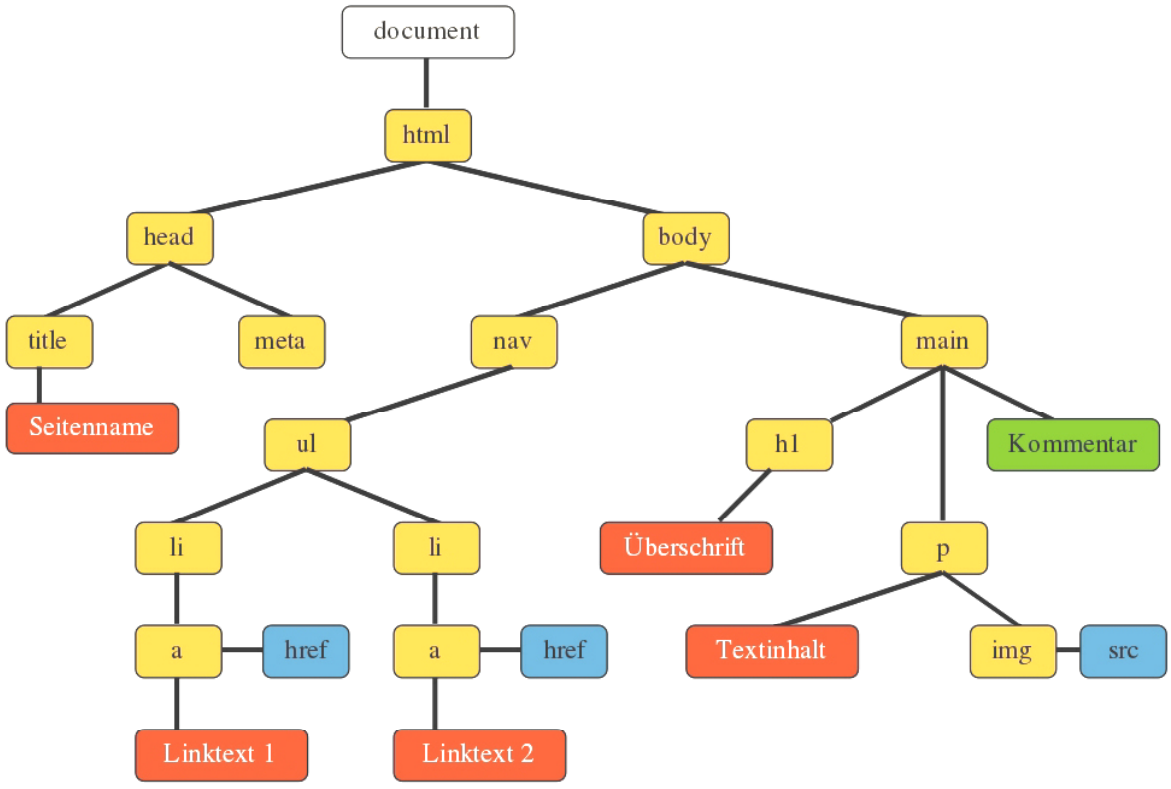
Ein Web-Browser auf dem Rechner des Benutzers (Client) stellt eine Verbindung mit dem Rechner des Anbieters (Server) über einen Internet-Link her und schickt an den Webserver eine Anfrage. Der Webserver antwortet mit einer Empfangsbestätigung und sendet an den Client ein HTML-Protokoll zurück.

Weil JavaScript im Web-Browser des Client integriert ist und auf dessen Rechner dann ausgeführt wird, ist JavaScript eine clientseitige Programmiersprache. Davon zu unterscheiden sind die serverseitigen Programmiersprachen (beispielsweise PHP), die nur am Server-Rechner laufen.

Der erhaltene HTML-Code liegt im Browser zunächst nur als Text vor. Während der Browser über das Netz den Code empfängt, wird durch einen so genannten Parser der Code schrittweise verarbeitet, d.h. in eine bestimmte Speicherstruktur im Arbeitsspeicher des Rechners übergeführt. Diese Speicherstruktur besteht aus zusammenhängenden Bündeln von Informationen, welche auch Objekte genannt werden. Die verschiedenen Objekte bilden eine Struktur, welche mit einem Wurzelknoten beginnt und sich dann baumartig zu weiteren Element-Knoten verzweigt. Der Aufbau dieser Objektstruktur ist durch das Document-Object-Model (DOM) geregelt. Das oberste Element (Wurzelknoten, root) trägt den Namen „***document***“, welches das HTML-Dokument im Browserfenster abbildet. Das „***document***“ enthält nun weitere Objekte: Zunächst „***HTML***“, „***HEAD***“, „***BODY***“, und dann beispielsweise Textfelder, Listen, Schaltflächen (buttons), Bilder (images), usw. Jedes Objekt besitzt bestimmte Eigenschaften (Attribute) und bestimmte Methoden (Funktionen). Die Attribute von übergeordneten Objekten werden auf unter-geordnete Objekte vererbt.

An den einzelnen Objekten kann der Benutzer bestimmte Ereignisse (events) auslösen, beispielsweise mit der Maus auf einen Schalter klicken (click) oder auf der Tastatur eine Taste drücken (keypress).

Die Programmiersprache JavaScript kann nun auf die verschiedenen Objekte zugreifen und auch mögliche Ereignisse registrieren und mit bestimmten Unterprogrammen (Funktionen) darauf reagieren. Diese Behandlung von Ereignissen (event handling) ermöglicht erst die Interaktion von Benutzer und HTML-Dokument.

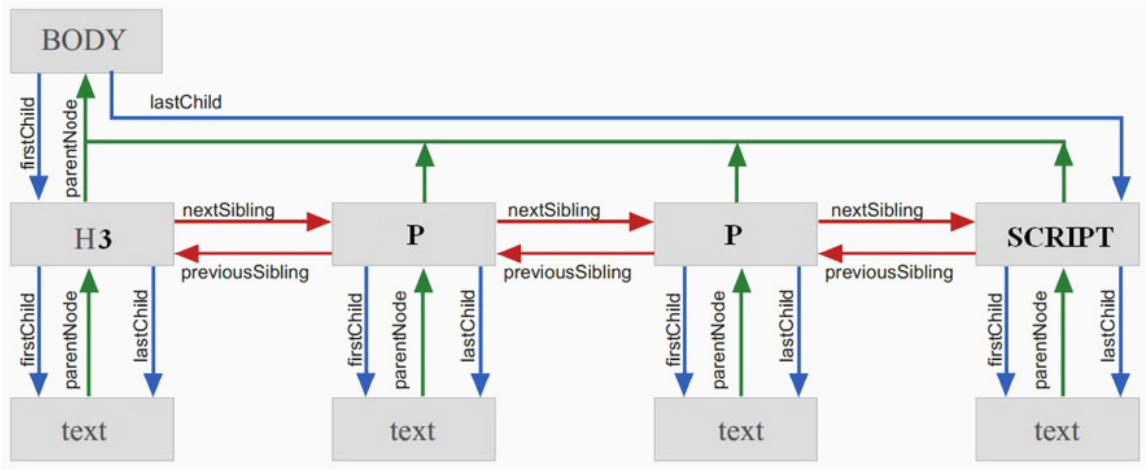


Schematische Darstellung eines DOM-Baumes mit HTML-Objekten.

Wie bereits beschrieben, besteht der DOM eines HTML-Dokuments aus unterschiedlichen, baumartig verzweigten Objekten, den so genannten Knoten (nodes). Das Wurzel-Element ist immer „**document**“. Direkt darunter liegen HTML, HEAD und BODY, und dann alle weiteren Elemente des aktuellen Dokuments, welche auch Siblings (Geschwister) genannt werden.

JavaScript ermöglicht mithilfe eigener Befehle den gezielten Zugriff auf alle DOM-Objekte. Mit *nextSibling* und *previousSibling*, *firstChild* und *lastChild* erreicht JavaScript jedes Element – auch Textknoten, Leerzeichen und Kommentare. Dazu kommen noch *parentNode*, *firstElementChild* und *lastElementChild*, *nextElementSibling* und *previousElementSibling*, welche auf Nodes zielen.

Von einem erreichten Knoten können *nodeType*, *nodeName* und *tagName* abgefragt werden. Der Knoteninhalt kann dann mit *innerHTML* oder *textContent* ausgelesen werden. Die untenstehende Grafik veranschaulicht beispielhaft einen DOM-Baum mit seinen Knoten und Zeigern.

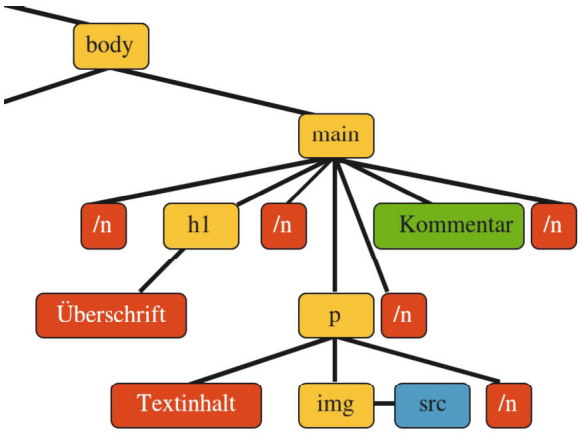


Das folgende Programm (***domscan.html***) durchläuft den gesamten DOM des aktuellen HTML-Dokuments. Dabei wird die rekursive Funktion ***domRunner*** verwendet, und bei jedem erreichten Knoten erfolgt die Anzeige seines Namens und des Namens seines Vorfahrens (parent).

```
<!DOCTYPE html>
<HTML>
<HEAD>
<title>domscan.html</title>
<meta charset="ISO-8859-1">
<meta name="description" content="Rekursive DOM-Darstellung">
<meta name="author" content="Paukert Herbert">
</HEAD>
<BODY>
  <h3>Dom-Scanner</h3>
  <div>
    <p>Das Programm durchläuft rekursiv den DOM !</p>
  </div>
  <script>
    function info(node) {
      var s = 'Node: ' + node.nodeName +
              '\nParent: ' + node.parentNode.nodeName;
      // eventuell: + '\nContent: ' + node.innerHTML;
      alert(s);
    }
    function domRunner(node,func) {
      func(node);
      node = node.firstElementChild;
      while (node) {
        domRunner(node,func);
        node = node.nextElementSibling;
      }
    }
    domRunner(document.querySelector("BODY"),info);
  </script>
</BODY>
</HTML>
```

Grundsätzlich gibt es drei verschiedene Knotentypen: Elementknoten – Textknoten – Attributknoten.
Das folgende Beispiel zeigt einen Ausschnitt aus einer HTML-Seite:

```
<main>
    <h1>Überschrift</h1>
     <!-- Hier ist ein Kommentar -->
     <p>
        Textinhalt
        <img> src="bild.jpg" alt="">
     </p>
</main>
```



Hinweis: Mittels *document.createElement()* und *document.removeElement()* können neue Objekte erzeugt oder schon bestehende aus dem DOM entfernt werden. Ausführliche Informationen dazu findet man auf der Buchseite 105.

Das window-Objekt von JavaScript

• Das *window*-Objekt

Es repräsentiert das aktuelle Webbrowser-Fenster mitsamt dem HTML-Dokument, den Dialogboxen, den Ereignissen und den Timern. Es enthält u.a. die beiden Attribute *window.innerWidth* und *window.innerHeight*, welche die Breite und Höhe des Fensters angeben und auch weitere Methoden zum Öffnen und Schließen von neuen Fenstern:

window.open("test.html","fenster01","width=640,height=480,scrollbars=yes");

Alle globalen Variablen und Funktionen liegen im *window*-Objekt. Ein direkter Nachkomme von *window* ist das *document*-Objekt, mit dessen Hilfe JavaScript mit dem DOM einer HTML-Seite kommuniziert.

• Das *window.screen*-Objekt

Es enthält Information über Auflösung, Höhe, Breite und Farbtiefe des Bildschirms.

• Das *window.location*-Objekt

Es enthält sämtliche relevanten Informationen über die aktuelle Webseite. *location.href* liefert die Internetadresse (URL = Uniform Ressource Locator), beispielsweise von http://www.paukert.at
(http = Hypertext Transfer Protocol, www = World Wide Web, Homepage-Name, Länderkennung).

• Zwei *timer*-Mehoden von *window*

Das *window*-Objekt ermöglicht es, einen Programmcode in ganz bestimmten Zeitintervallen auszuführen. Hierzu werden folgende zwei vordefinierte Methoden angeboten:

var tvar = window.setInterval(func,zeit): Führt die Funktion *func* immer nach Zeitintervallen *zeit* (in Millisekunden) aus. Die Methode *window.clearInterval(tvar)* stoppt dann diese unendlich wiederkehrende Ausführung.

var tvar = window.setTimeout(func,zeit): Führt die Funktion *func* nur einmalig nach dem Zeitintervall *zeit* (in Millisekunden) aus. Soll die u.U. noch nicht ausgeführte Funktion abgebrochen werden, dann wird das mit *window.clearTimeout(tvar)* erreicht.

Das document-Objekt von JavaScript

Ein direkter Nachkomme des *window*-Objekts ist das *document*-Objekt, mit dessen Hilfe JavaScript mit dem DOM einer HTML-Seite kommuniziert. Die folgenden Buchseiten liefern ausführliche Erklärungen und Beispiele dafür, wie JavaScript auf HTML-Objekte zugreift und diese verändert. Die vorliegende Seite bietet nur einen kurzen Überblick.

• Liste von einigen Attributen und Methoden

document.URL	Liefert die volle URL des Dokuments
document.title	Liefert den Titel des Dokuments
document.images	Liefert eine Sammlung von allen <img> Elementen des Dokuments
document.createElement()	Erzeugt einen Element-Knoten
document.createTextNode()	Erzeugt einen Text-Knoten
el1.appendChild(*el2*)	Erzeugt ein Kind-Element *el2* des Vater-Elements *el1*
el1.removeChild(*el2*)	Entfernt ein Kind-Element *el2* des Vater-Elements *el1*
getElementById(*'value'*)	Liefert das Element, dessen ID-Attribut den spezifischen Wert *'value'* hat
addEventListener()	Heftet einen Event-Handler an das Dokument
removeEventListener()	Entfernt einen Event-Handler aus dem Dokument

• Verändern von Inhalt und CSS-Style eines Elements (siehe Beispiel)

el = getElementById(*'ElemId'*);	Liefert ein Element mit dem Identitätsattribut *ElemId*
el.innerHTML = *'Inhalt'*;	Ändert mittels *innerHTML* den Element-Inhalt *Inhalt*
el.style.*property* = *'Wert'*;	Ändert mittels *CSS-Style* das Element-Attribut *property* (z.B. el.style.backgroundColor = *'darkblue'*;)

• Einrichten von Event-Handlern (siehe Beispiel)

Bei jeden Event (Ereignis) müssen das **Objekt** (z.B. button), der **Typ** (z.B. mousedown) und die jeweilige **Verarbeitungs-Funktion** (Event-Handler) unterschieden werden.
Es gibt **Fenster-Events** (load, scroll, unload, …), **Maus**-Events (click, dblclick, mousedown, mousemove, mouseup), **Tastatur-Events** (keypress, keydown, keyup), **Formular-Events** (change, focus, submit, reset).
Für das Einrichten von Event-Handlern gibt es 3 Möglichkeiten:

(1) Event-Handler als HTML-Attribut	(im Beispiel bei Button „btn1“)
(2) Event-Handler als Objekt-Methode	(im Beispiel bei Button „btn2“)
(3) Event-Handler mit „addEventListener“	(im Beispiel bei Button „btn3“)

• Beispiel (*events.html*):

```
<!doctype html>
<html>
<head>
</head>
<body>
<p id='info'>Ich bin ein Paragraph.</p>
<button id='btn1' onclick='changeColor()'> (1) Farbe ändern </button><br>
<button id='btn2'> (2) Text ändern </button><br>
<button id='btn3'> (3) Farbe und Text ändern </button><br>
<script>
   var change = false;
   function changeColor() {
     change = !change;
     if (change) {farbe = 'red'} else {farbe = 'blue'}
     document.getElementById('info').style.color = farbe;
   }
   function changeText() {
     change = !change;
     if (change) {text = 'Ich bin ein Paragraph.'} else {text = 'Ich bin ein Absatz.'}
     document.getElementById('info').innerHTML = text;
   }
   function changeAll() { change = !change; changeColor(); changeText(); }

   document.getElementById('btn2').onclick = changeText;
   document.getElementById('btn3').addEventListener('click',changeAll);
</script>
</body>
</html>
```

Vordefinierte und benutzerdefinierte Objekte

In JavaScript muss zwischen vordefinierten und benutzerdefinierten Objekten unterschieden werden. Wie vom Benutzer Objekte konstruiert und verwendet werden, ist im Teil 1 dieses Lehrbuches ausführlich erklärt. Hier soll nur eine kurze Übersicht über die wichtigsten vordefinierten Objekte gegeben werden. Diese Objekte können entweder in der „Konstruktor"-Schreibweise mit dem Schlüsselwort „new" oder in der „Literal"-Schreibweise nur mit den typ-spezifischen Klammern definiert werden. Der Konstruktor hat den Vorteil, dass damit ein leeres Basisobjekt erzeugt wird, von dem dann auf einfache Weise beliebige Instanzen erzeugt werden können.

(1) Für die primitiven (einfachen) Datentypen ***String, Number*** und ***Boolean*** stellt JavaScript ebenfalls entsprechende Objekte zur Verfügung, in welche der Datentyp automatisch umgewandelt wird, wenn man eine der vielen einschlägigen Methoden des Datentyps aufruft.

(2) Das ***object*–Objekt**. Beispiel:

(a)
```
function Person(name, alter) {
    this.x = name;
    this.y = alter;
    this.ausgeben = function() {
        info = this.x + ", " + this.y;
        alert(info);
    }
var kunde01 = new Person("Meier",55);
kunde01.ausgeben();
```

(b)
```
var Person = { name: "Meier", alter: 55 }
```

(3) Das ***array*-Objekt**. Beispiel:

(a)
```
var ort = new Array();
ort[0] = "Berlin";
ort[1] = "Paris";
ort[2] = "Wien";
```

(b)
```
var ort = ["Berlin", "Paris", "Wien"];
```

(c) Einschlägige Methoden, wie beispielsweise *ort.sort();*

(4) Das ***function*-Objekt**: *function*(a,b,…) { ……. }
Ein Konstruktor *new function();* muss hier nicht verwendet werden.

(5) Das ***Date*-Objekt**. Beispiel: var *datum = new Date()*; alert(*datum.toDateString()*);
Dieses vordefinierte Objekt stellt viele Methoden für unterschiedliche Berechnungen zur Verfügung.

(6) Das ***Math*-Objekt**. Beispiel: var *r = 5*; var a = *Math.pow(r,2) * Math.PI*;
Ein Konstruktor *new Math();* muss hier nicht verwendet werden.

(7) Das ***RegExp*-Objekt**. Damit werden so genannte reguläre Sprachausdrücke erzeugt, wobei es sich um String-Objekte handelt, welche nach einer speziellen Syntax auf gebaut sind. Sie werden zum Suchen (search) und Ersetzen (replace) verwendet.

Ein Beispiel: var *str* = "In genau diesem Text wird jetzt gesucht.";

(a) var posi = str.**search**(/wird/); // reguläre Textsuche (/ = Begrenzung)

(b) var strneu = str.**replace**(/e/gi, 'XYZ'); // reguläre Textersetzung

Hier wird überall (global, g) im Text 'e' durch 'XYZ' ersetzt und zwar case-insensitiv (i).

(c) var count = str.**match**(/e/gi).length; // reguläre Textzählung (wie oft 'e' vorkommt)

(d)
```
var reg = new RegExp(/Text/);          // definiert ein neues RegExp-Objekt
var posi = str.search(reg);            // ergibt Position 16
var strneu = str.replace(reg, 'Absatz'); // ersetzt 'Text' durch 'Absatz'
```

[2.2] Zugriff auf HTML-Objekte

Grundsätzlich kann JavaScript erstens mithilfe des DOM den Inhalt von einzelnen HTML-Objekten verändern und dann zweitens mithilfe von CSS-Styles die Präsentation der Objekte verändern. Drittens werden diese Manipulationen über Maus-Klicks auf einen Button ausgelöst. Das Alles ist nur möglich, wenn im HTML-Dokument den Objekten eine so genannte Identitätskennung (ID) zugewiesen wird.

```
<!DOCTYPE html>
<html>
<head>
   <title> DOM-Schnittstelle </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
</head>

<body>
<h3> Die Schnittstelle zum DOM </h3>
<p id="info"> Der Absatztext </p>
<script>
   var text = document.getElementById("info").innerHTML;
   text = text + " wurde erweitert ! ";
   document.getElementById("info").innerHTML = text;
</script>
<body>
<html>
```

In diesem Programm wird einem Absatz das Attribut ***id ="info"*** zugewiesen. Zuerst wird der Inhalt (content) des Absatzes mit der Anweisung ***document.getElementById("info").innerHTML*** auf der Variablen ***text*** abgespeichert. Dann erfolgt eine Erweiterung des Textes und zum Schluss wird der so veränderte Text wieder als neuer Inhalt in den Absatz geschrieben.

Der Transfer des Inhaltes eines HTML-Elementes auf eine JavaScript-Variable und auch wieder zurück erfolgt mit der Methode ***getElementById()*** und der zusätzlichen Eigenschaft ***innerHTML*** des obersten DOM-Objektes ***document***.

In der nachfolgenden Programmvariation wird nach der Existenz eines HTML-Objekts gefragt.

```
<script>
   var elem = document.getElementById("info");
   if ( elem ) {
      var text = elem.innerHTML;
      text = text + " wurde erweitert ! ";
      elem.innerHTML = text;
   }
   else { alert("Das Element mit der ID info wurde nicht gefunden! "); }
</script>
```

Eine andere Methode ***var elem = getElementsByName("xyz")*** greift auf das erste HTML-Objekt zu, das mit dem Namensattribut ***"xyz"*** versehen ist. Wie mit ***<input name= "xyz" type="text" value="Herbert">*** kann der Variablen ***elem*** der String ***"Herbert"*** zugewiesen werden (mit *elem.innerHTML = "Herbert"***).**

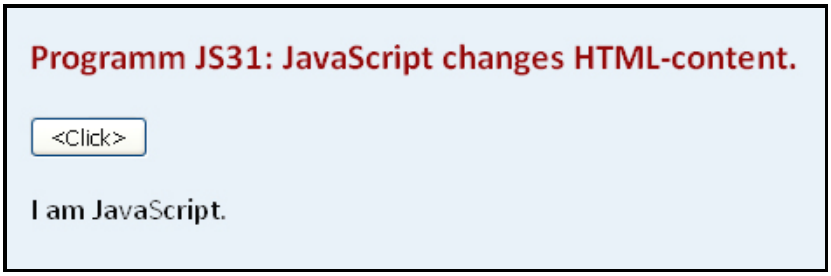
Die hier beschriebenen Zugriffsmethoden auf HTML-Objekte sind die häufigsten. Daneben gibt es aber weitere spezifische Methoden wie *getElementsByTagName()* und *getElementsByClassName()*, oder *querySelektor()* und *querySelektorAll()*. Siehe dazu auch Buchseite 105.

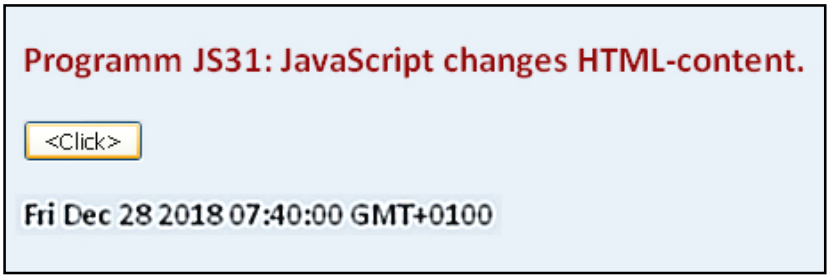
Mit ***document.getElementById("info").style.color = "red"*** wird nicht auf den Inhalt (content), sondern auf ein Attribut eines HTML-Elementes zugegriffen. Als Besonderheit gilt, dass CSS-Eigenschaften, die einen Bindestrich (-) enthalten in JavaScript ohne diesen Bindestrich geschrieben werden. Außerdem wird dann der erste Buchstabe nach dem Bindestrich groß geschrieben, d.h. statt *font-style* wird *fontStyle* geschrieben: ***document.getElementById("info").style.fontStyle = "italic"***.

Ein Spezialfall liegt bei Formularen vor, wo mit ***document.Formularname.Elementname.value*** auf den Wert des Elementes zugegriffen wird.

[2.2.1] Verändern des Inhalts von HTML-Objekten

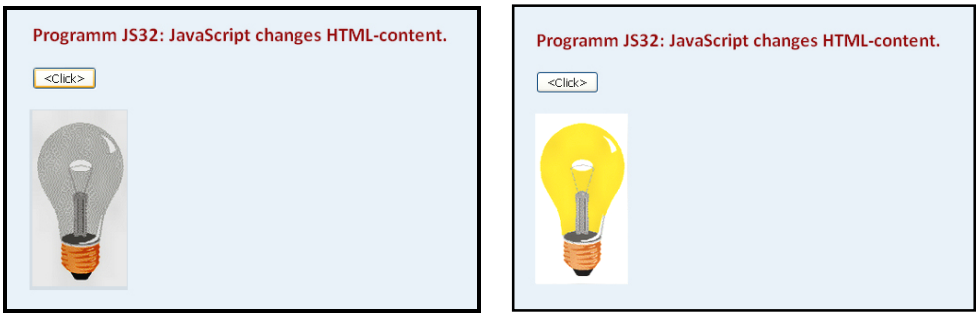
Im folgenden Programm wird abwechselnd der Absatzinhalt „*I am JavaScript*“ durch das aktuelle Datum *Date()* ersetzt.






```
<!DOCTYPE html>
<html>
<head>
   <title> JS31 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style type="text/css">
       body { background-color: #E0E8EF; color: black;
              font-family: Calibri; font-size: 18px; font-weight: normal;
              text-size-adjust: none; text-align: left; margin: 5%; }
       h3 { color: darkred; }
   </style>
   <script>
      var original = true;
      function func() {
         original = !original;
         var tex1 = "I am JavaScript.";
         var tex2 = Date();
         if (original) { document.getElementById("demo").innerHTML = tex1; }
         else { document.getElementById("demo").innerHTML = tex2; }
      }
   </script>
</head>

<body>
   <h3> Programm JS31: JavaScript changes HTML-content. </h3>
   <input type="button" value="<Click>" onclick="func();">
   <br>
   <p id="demo">I am JavaScript.</p>
</body>
</html>
```

Im folgenden Programm werden einem Image-Objekt abwechselnd
zwei verschiedene Bildquellen zugewiesen.




```
<!DOCTYPE html>
<html>
<head>
   <title> JS32 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style type="text/css">
       body { background-color: #E0E8EF; color: black;
              font-family: Calibri; font-size: 18px; font-weight: normal;
              text-size-adjust: none; text-align: left; margin: 5%; }
       h3 { color: darkred; }
   </style>
   <script>
      var original = true;
      function func() {
         original = !original;
         if (original) { document.getElementById("demo").src = "lamp_off.jpg"; }
         else { document.getElementById("demo").src = "lamp_on.jpg"; }
      }
   </script>
</head>

<body>
   <h3> Programm JS32: JavaScript changes HTML-content.</h3>
   <input type="button" value="<Click>" onclick="func();">
   <br>
   <br>
   <image id="demo" src="lamp_off.jpg" style="width:100px" alt="KEINE Lampen vorhanden!">
</body>
</html>
```

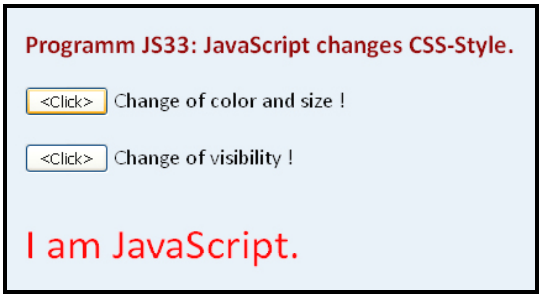
[2.2.2] Verändern des CSS-Styles von HTML-Objekten

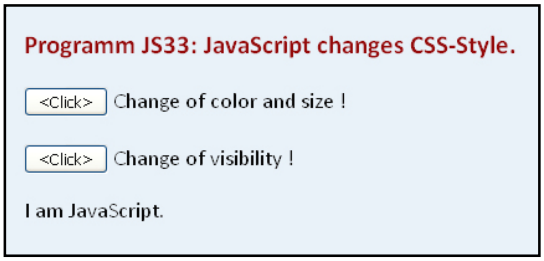
Jedes HTML-Objekt „element“ besitzt ein „style“-Attribut, das selbst wieder ein Objekt ist.
Mit JavaScript kann auf die verschiedenen CSS-Eigenschaften von diesem „style“-Objekt
zugegriffen werden: **element.style.CSS-Eigenschaft = "Wert";**

Dazu ein Beispiel:

```
   <p id = "absatz"> Ich bin ein einfacher Absatz.</p>

   <script>
      var element = document.getElementById("absatz");
      element.style.color = "red";
      element.style.fontSize = "18px";
   </script>
```

Diese Direktformatierung über das „style“-Objekt wird auch „Inline-Styles“ genannt, und es gilt dabei die so genannte **CamelCase**-Regel: Bei allen CSS-Eigenschaften, welche einen Bindestrich enthalten, wird dieser weggelassen und der Buchstabe nach dem Bindestrich wird groß geschrieben, was an einen Kamel-Buckel erinnern soll. So muss statt „font-size“ dann „fontSize“ geschrieben werden.

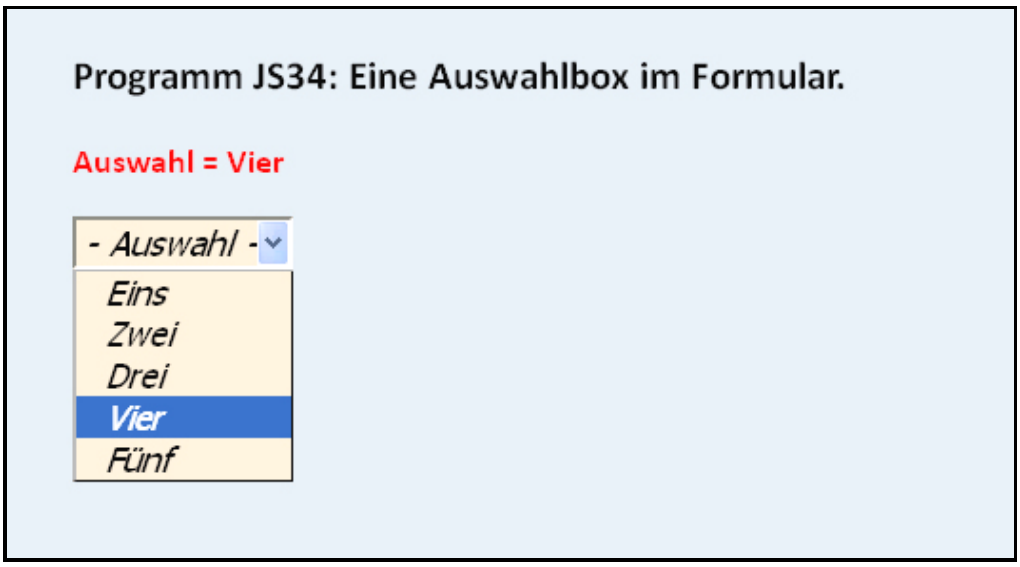





```
<!DOCTYPE html>
<html>
<head>
   <title> JS33 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style type="text/css">
       body { background-color: #E0E8EF; color: black;
              font-family: Calibri; font-size: 18px; font-weight: normal;
              text-size-adjust: none; text-align: left; margin: 5%; }
       h3 { color: darkred; }
   </style>
   <script>

      var original = true;
      var sichtbar = true;

      function func1() {
         var el = document.getElementById("demo");
         original = !original;
         if (original) {
            el.style.fontSize="18px";
            el.style.color="black";
         }
         else {
            el.style.fontSize="36px";
            el.style.color="red";
         }
      }

      function func2() {
         var el = document.getElementById("demo");
         sichtbar = !sichtbar;
         if (sichtbar) { el.style.visibility = "visible"; }
         else { el.style.visibility = "hidden"; }
      }

   </script>
</head>

<body>
   <h3> Programm JS33: JavaScript changes CSS-Style.</h3>
   <input type="button" value="<Click>" onclick="func1();"> Change of color and size !
   <br>
   <br>
   <input type="button" value="<Click>" onClick="func2();"> Change of visibility !
   <br>
   <p id="demo">I am JavaScript.</p>
</body>
</html>
```

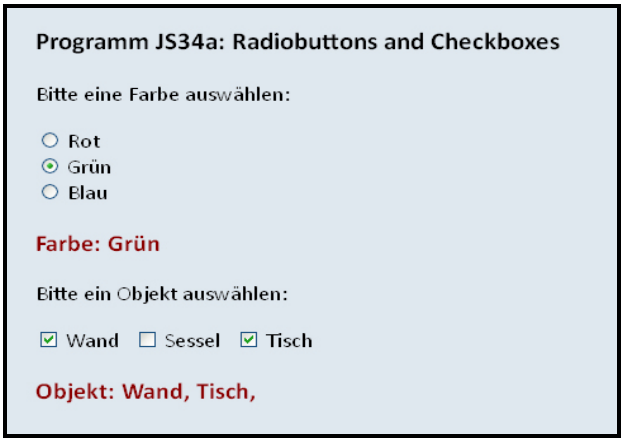
[2.2.3] Zugriff auf die Items einer Auswahlbox

Im folgenden Programm wird auf die einzelnen Items einer Auswahlbox zugegriffen.



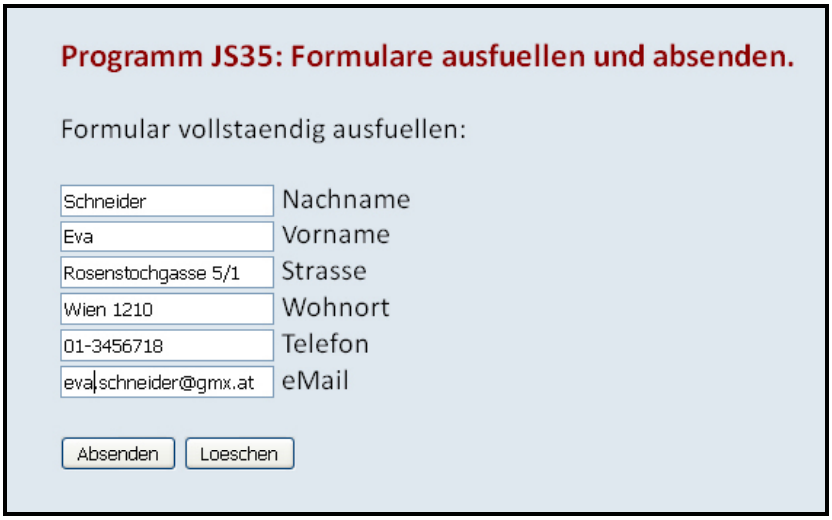

```
<!doctype html>
<html>

<head>
   <title> JS34 </title>
   <meta charset="ISO-8859-1">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta name="author" content="Herbert Paukert">
   <style>
      body  { background-color:#E0E8EF; color:black; margin:5%;
              font-family:Calibri; font-size:18px; font-weight:normal; }
      #sbox { background-color:antiquewhite; font-size:18px; font-style:italic; }
      #info { color:red; font-weight:bold; }
      h3    { color:darkred; }
   </style>
   <script>
      function func(element)
      {
        var eingabe = element[element.selectedIndex].text.trim();
        if (element.selectedIndex == 3 ) { eingabe = 'Ich bin der Durchschnitt!'; }
        document.getElementById("info").innerHTML = 'Auswahl = ' + eingabe;
        element.value = '- Auswahl -';
      }
   </script>
</head>

<body>
<form>
   <h3>Programm JS34: Eine Auswahlbox im Formular.</h3>
   <p id="info">Auswahl = ?</p>
   <select name="liste" id="sbox" size="1" onchange="func(this.form.liste)">
      <option selected disabled hidden>- Auswahl -</option>
      <option>  Eins  </option>
      <option>  Zwei  </option>
      <option>  Drei  </option>
      <option>  Vier  </option>
      <option>  Fünf  </option>
   </select>
 </form>
</body>

</html>
```

[2.2.4] Zugriff auf Radiobuttons und Checkboxes

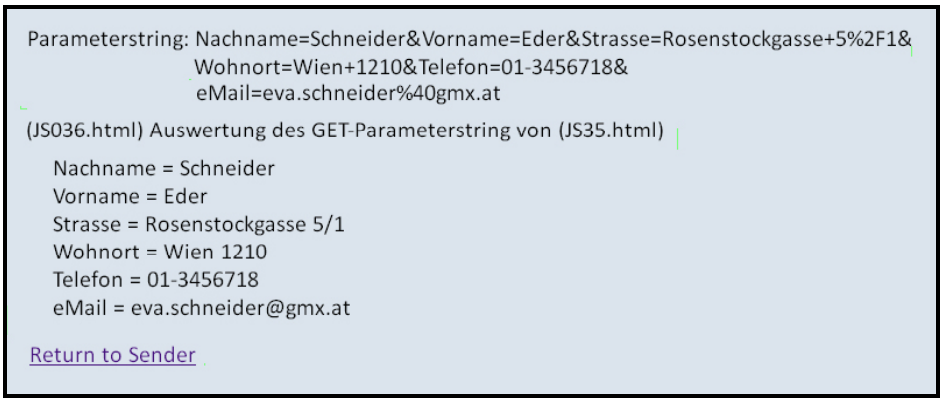



```
<!DOCTYPE html>
<html>
<head>
<title>JS34a</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="author" content="Herbert Paukert">
<style>
   body {background-color: #E0E8EF; color: black;
         font-family: Calibri; font-size:18px; margin: 2%; }
   .erg {color: darkred; font-size:20px; font-weight: bold; }
</style>
</head>

<body>
  <form>
  <h3>Programm JS34a: Radiobuttons and Checkboxes</h3>
  <p>Bitte eine Farbe auswählen:</p>
  <p>
     <input type="radio" name="col" value="Rot" checked> Rot <br>
     <input type="radio" name="col" value="Grün"> Grün <br>
     <input type="radio" name="col" value="Blau"> Blau <br>
  </p>
  <p id="color" class="erg">Farbe:</p>

  <p>Bitte ein Objekt auswählen:</p>
  <p>
     <input type="checkbox" name="obj" value="Wand" checked id="c1"> Wand  
     <input type="checkbox" name="obj" value="Sessel" id="c2"> Sessel  
     <input type="checkbox" name="obj" value="Tisch" id="c3"> Tisch  
   </p>
   <p id="obj" class="erg">Objekt:</p>
  </form>

  <script>
    var farbe = document.getElementsByName("col");
    for (var i = 0; i < farbe.length; i++) {
      farbe[i].onclick = function() {
         let txt = this.value;
         document.getElementById("color").innerHTML = 'Farbe: ' + txt;
      }
    }

    var objekt = document.getElementsByName("obj");
    for (var i = 0; i < objekt.length; i++) {
      objekt[i].onclick = function() {
         var txt = "Objekt: ";
         for (var j = 0; j < objekt.length; j++) {
           if (objekt[j].checked) {
              txt = txt + objekt[j].value + ', ';
           }
           document.getElementById("obj").innerHTML = txt;
         }
       }
    }
  </script>

</body>
</html>
```

[2.2.5] Formulare gesichert ausfüllen, absenden und empfangen




```
<!DOCTYPE html>
<html>
<head>
   <title> JS35 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style type="text/css">
       body { background-color: #E0E8EF; color: black;
              font-family: Calibri; font-size: 20px; font-weight: normal;
              text-size-adjust: none; text-align:left; margin:5%; }
       h3 { color: darkred; }
   </style>
   <script>
      function pruef() {
         var formular = document.ein;
         var len = formular.length;
         for (i = 0; i < len; i++) {
             var feld = formular.elements[i];
             var txt = feld.value;
             if (txt == "") {
                formular.elements[i].focus();
                return false;
             }
         }
      }
</script>
</head>

<body>
   <h3> Programm JS35: Formulare ausfuellen und absenden. </h3>
    Formular vollstaendig ausfuellen:
   <br><br>

   <!--
   <form name="ein" action="mailto:..." method="post" onsubmit="return pruef()">
   -->
   <form name="ein" action="js36.html" method="get" onsubmit="return pruef()">
   <input type="text" name="Nachname"> Nachname <br>
   <input type="text" name="Vorname"> Vorname <br>
   <input type="text" name="Strasse"> Strasse <br>
   <input type="text" name="Wohnort"> Wohnort <br>
   <input type="text" name="Telefon"> Telefon <br>
   <input type="text" name="eMail"> eMail <br>
   <br>
   <input type="submit" value="Absenden">
   <input type="reset" value="Loeschen">
   <br>
   </form>

   <script> document.forms[0].elements[0].focus(); </script>
</body>
</html>
```




```
<!doctype html>
<html>
<head>
   <title> JS36 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style type="text/css">
       body{background-color:#E0E8EF; color:black;
       font-family:Calibri,sans-serif; font-size:20px; text-align:left; margin:5%;}
   </style>
</head>
<body>
<p id="info">Parameterstring: </p>
<script>
  // Decodierung von Sonderzeichen im GET-Parameterstring
  with (document) {
    write("<br>(JS036.html) Auswertung des GET-Parameterstring von (JS35.html)<br><br>");
    var x = location.search.substring(1);
    document.getElementById("info").innerHTML = 'Parameterstring: ' + x;
    var c = "";
    var y = "";
    for (var i = 0; i < x.length; i++) {
      c = x.charAt(i);
      if (c == "+") { c = " "; }
      y = y + c;
    }
    x = y;
    x = x.replace(/%21/g,"!");
    x = x.replace(/%22/g,'"');
    x = x.replace(/%23/g,'#');
    x = x.replace(/%24/g,"$");
    x = x.replace(/%25/g,"%");
    x = x.replace(/%28/g,"(");
    x = x.replace(/%29/g,")");
    x = x.replace(/%2A/g,"*");
    x = x.replace(/%2B/g,"+");
    x = x.replace(/%2C/g,",");
    x = x.replace(/%2D/g,"-");
    x = x.replace(/%2F/g,"/");
    x = x.replace(/%3F/g,"?");
    x = x.replace(/%40/g,"@");
    x = x.replace(/%A7/g,"§");
    x = x.replace(/%C4/g,"Ä");
    x = x.replace(/%E4/g,"ä");
    x = x.replace(/%D6/g,"Ö");
    x = x.replace(/%F6/g,"ö");
    x = x.replace(/%DC/g,"Ü");
    x = x.replace(/%FC/g,"ü");
    x = x.replace(/%DF/g,"ß");
    x = x.replace(/=/g," = ");
    var z = x.split("&");
    for (var i = 0; i < z.length; i++)  {
      write("  " + z[i] + "<br>");
    }
  }
</script>
<br><br>
  <a href="js35.html">Return to Sender</a>
<br><br>
</body>
</html>
```

[2.2.6] Verschiedene Event-Handler einrichten

An den verschiedenen HTML-Elementen kann der Anwender unterschiedliche Ereignisse (Events) auslösen, beispielsweise Maus-Ereignisse (click, mousedown, mouseover, mouseup), Tastatur-Ereignisse (keydown, keyup, keypress), Fenster-Ereignisse (load, unload, reset, resize, …), usw.

Nach dem Laden einer HTML-Seite kann mit einem JavaScript-Programm ein vom Anwender ausgelöstes Ereignis registriert und darauf mit einer Ereignis-Verarbeitungs-Funktion (Event Handler) asynchron reagiert werden. „Asynchron" bedeutet dabei, dass auch nach Ablauf und Beendigung des Skripts der Handler zur Verfügung steht. Bei jedem Event-Handling wird ein so genanntes **Event-Objekt** erzeugt, welches verschiedene Eigenschaften und Methoden enthält, die im Handler verwendet werden können. Dieses Event-Objekt wird der Handler-Funktion automatisch immer als erster Parameter übergeben und sein Bezeichner ist frei wählbar (meistens „ev" oder nur „e").

Jede Ereignis-Überwachung besteht aus drei Teilen: Erstens das HTML-Element (z.B. „button"), zweitens der Ereignis-Typ (z.B. „click") und drittens die Ereignis-Verarbeitungs-Funktion (Handler). Der Aufruf des Handlers erfolgt dann beispielsweise mit „button.**on**cklick" oder „window.**on**load".

Beim **traditionellen Event-Handling** gibt es zwei Möglichkeiten: Entweder wird das Ereignis direkt am HTML-Element registriert (inline-Handling), oder die Ereignis-Registrierung erfolgt im Skript.

Beispiel 1:

```
<script>
   function Antwort(ev) { alert("Danke, ich wurde angeklickt mittels [" + ev.type + "]"); }
</script>
 <p onclick = "Antwort(event)"> Bitte, klicken Sie mich an !</p>
```

Beispiel 2:

```
<script>
   window.onload = start;
   function start() {
     document.getElementById("absatz").onclick = Antwort();
   }
   function Antwort() {
     var text = "Danke, ich wurde angeklickt !";
     document.getElementById("absatz").innerHTML = text;
   }
</script>
 <p id="absatz"> </p>
```

Beim **modernen Event-Handling** wird ein so genannter „**addEventListener**" aufgerufen, der bei jedem HTML-Element als Ereignis-Überwachungs-Methode automatisch vorhanden ist.

element.addEventListener(event-Typ, event-Handler, capturing);

Die Wirkungsweise der beiden ersten Parameter ist offenkundig. Der dritte Parameter ist optional und kann „true" oder „false" sein. Bei „false" oder leer wird der DOM-Elementebaum beginned beim Zielelement aufsteigend bis zum Wurzelelement „window" durchklettert, und dabei kann jedesmal der Element-Handler des Zielelementes ausgeführt werden. Dieser Event-Fluss („bubbling") ist Standard. Bei „true" hingegen verläuft der Event-Fluss umgekehrt von aussen nach innen („capturing"). Der Event-Fluss wird auch als Ereignis-Ausbreitung (Event-Propagation) bezeichnet, und er kann mittels „event.stopPropagation" unterbunden werden.

Das nachfolgende fortschrittliche Beispiel 3 entspricht dem traditionellen Beispiel 1.

Beispiel 3:

```
<script>
   document.addEventListener("click", function() {
      document.getElementById("absatz").onclick = Antwort();
   },true);
   function Antwort() {
     var text = "Danke, ich wurde von der Maus angeklickt !";
     document.getElementById("absatz").innerHTML = text;
   }
</script>
<p id="absatz"> Bitte, klicken Sie mich an !</p>
```

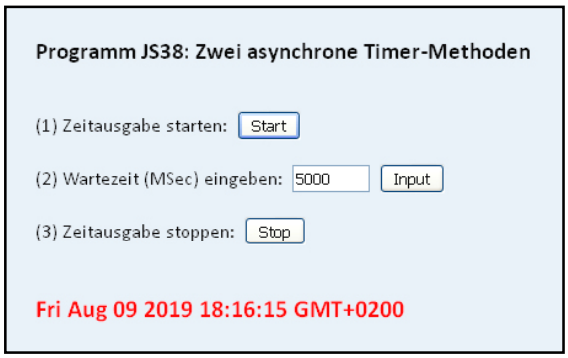

```
<!doctype html>
<html>
<head>
   <title> JS37 (Event-Handling) </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style>
      body { background-color:#E0E8EF; color:black;
             font-family:Calibri; font-size:18px;
             margin:5%; }
   </style>

   <script>

     var original = false;

     function changeColor() {
        original = !original;
        if ( original ) {
           document.getElementById("absatz").style.color = "red";
           document.getElementById("absatz").style.fontStyle = "italic";
        }
        else {
           document.getElementById("absatz").style.color = "black";
           document.getElementById("absatz").style.fontStyle = "";
        }
     }

//   (1) Traditionelles Event-Handling für ein "Fenster"-Ereignis
//   als ein, dem "window" zugeordneter onload-Handler.
//   (immer OHNE Parameter-Klammern aufrufen !)

     window.onload = start;

//   (2) Fortschrittliches Event-Handling für das "Fenster"-Ereignis

     document.addEventListener("DOMContentLoaded", function() {
       start();
     } );

     function start() { alert('Ich bin hier !'); }

//   Traditionelles Event-Handling für ein "Tastatur"-Ereignis
//   als ein, dem "body"-Element direkt angehängter onkeydown-Handler.
//   Zugriff auf ein Attribut des Event-Objektes "ev" mit "ev.keyCode"
//   oder mit "ev.which" (nur für Internet-Explorer)

     function getKey(ev) {
        var taste = ev.keyCode || ev.which;
        if (taste) {changeColor();}
        alert('Tastencode = ' + taste);
     }

//   Modernes Event-Handling für ein "mouseover"-Ereignis
//   als eine, dem "document" automatisch zugeordnete Callback-Funktion.

     document.addEventListener("mouseover", function() {
       document.getElementById("absatz").onmouseover = changeColor();
     });

   </script>
</head>

<body onkeydown = "getKey(event)">

   <font color = "darkred">
   <h3> Programm JS37: Verschiedene Event-Handler</h3>
   (1) mit -mouseOver-  auf dem div-Objekt<br>
   (2) mit -mouseClick- auf das Button-Objekt<br>
   (3) mit -keyPressed- auf der Leertaste<br>
   </font>
   <br><br>

   <!-- Traditioneller Schalter-Mausklick-Handler -->
   <button id="btn" onclick = "changeColor()"> Click Button </button>
   <br><br>

   <div id="absatz">
    ------------------<br>
    I am JavaScript.<br>
    I am JavaScript.<br>
    I am JavaScript.<br>
    ------------------<br>
   </div>

</body>
</html>
```

[2.2.7] Zwei TIMER-Methoden des WINDOW-Objektes

***var tvar = window.setInterval(func,zeit)*:** Führt die Funktion *func* immer nach Zeitintervallen *zeit* (in Millisekunden) aus. Die Methode *window.clearInterval(tvar)* stoppt dann diese unendlich wiederkehrende Ausführung.

***var tvar = window.setTimeout(func,zeit)*:** Führt die Funktion *func* nur einmalig nach dem Zeitintervall *zeit* (in Millisekunden) aus. Soll die u.U. noch nicht ausgeführte Funktion abgebrochen werden, dann wird das mit *window.clearTimeout(tvar)* erreicht.

```
<!doctype html>
<html>
<head>
<title> JS38 </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Herbert Paukert">

<style type="text/css">
   body { background-color:#E0E8EF; color:black;
          font-family:Calibri; font-size:16px; font-weight:normal;
          text-size-adjust: none; text-align:left; margin-left:5%; }
   p { color:red; font-size:20px; font-weight:bold; }
</style>

<script>
   var zeit = 5000;
   var run;

   function showText() {
     document.getElementById("info").innerHTML = 'Die Zeit vergeht ... ';
     document.getElementById("eingabe").value = zeit;
   }

   function getZeit(wert) {
     zeit = wert;
     alert(zeit);
   }

   function func() {
     var el = document.getElementById("info");
     var tex = Date();
     el.innerHTML = tex;
   }

   function startShow() {
      run = setInterval(func,1000);
   }

   function stopShow() {
      var el = document.getElementById("info");
      clearInterval(run);
      el.innerHTML = 'Aktion gestoppt. Noch ' + zeit + ' MSec warten !';
      setTimeout( function() {
        func(); alert(' Jetzt sind ' + zeit + ' MSec vergangen ! '); }, zeit);
      }
</script>
</head>

<body onload=showText();>
<form>
<br>
<h3> Programm JS38: Zwei asynchrone Timer-Methoden</h3>
<br>
(1) Zeitausgabe starten: 
<input type = "button" value = " Start " onclick = "startShow()">
<br><br>
(2) Wartezeit (MSec) eingeben:  <input type="text" id="eingabe" value="5000" size="6" > 
<input type = "button" value = " Input " onclick = "getZeit(this.form.eingabe.value)">
<br><br>
(3) Zeitausgabe stoppen: 
<input type = "button" value = " Stop " onclick = "stopShow()">
<br><br>
<p id="info"</p>
</form>
</body>
</html>
```



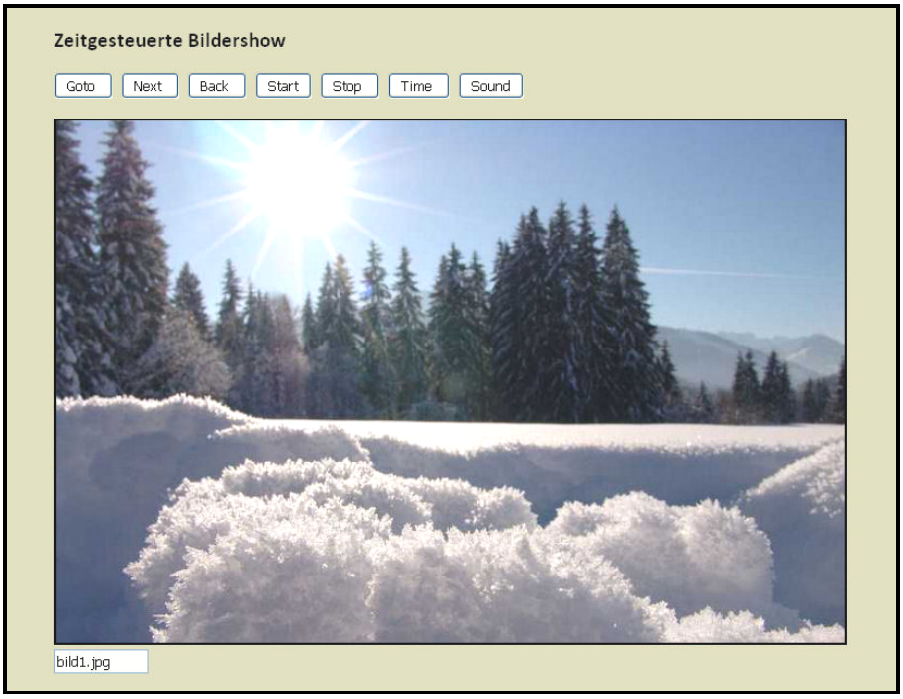
[2.3] Verwendung von MULTIMEDIA-Objekten

Multimedia ist die Präsentation von Texten, Bildern, Sounds und Videos.

Das Programm „js40.html“ realisiert eine Tabelle von Bildern (JPEG-Grafiken). Die Tabelle und die eingebetteten Bilder werden dynamisch erzeugt mit Hilfe der Anweisungen „**createElement**“, „**appendChild**“ und „**setAttribute**“.

Wird ein Bild angeklickt, dann öffnet die Funktion „**PopUp(bildname)**“ ein neues Fenster (window), in welchem das Bild dargestellt wird. Nach „**with(objekt)**“ folgt ein Code-Block mit Anweisungen, die sich alle auf das gleiche Objekt beziehen.

Das Programm „js41.html“ realisiert eine einfache Bildershow. Dabei kann der Bildwechsel entweder mittels Mausklick auf die Schalter (<Goto>, <Next>, <Back>) erfolgen oder mit Hilfe einer zeitgesteuerten Slideshow (<Start> und <Stop>). Außerdem kann die Zeitdauer der Bildanzeige mittels <Time> eingegeben werden. Mit <Sound> wird eine Musik ein-/ausgeschaltet.



[2.3.1] Bildergalerie

```
<!DOCTYPE html">
<html>
<head>
<title>js40.html</title>
<meta charset="ISO-8859-1">
<meta name="author" content="Herbert Paukert">
<meta name="description" content="Bilder">
<style>
  body {background-color: #444; color: #FFF; padding:1%; }
  .tab {background-color:antiquewhite; color:black; border:2px solid gray; padding:2px; }
</style>

<script>
   var max = 8;
   var name = 'bild';
   bild = new Array(max+1);
   for (var i = 1; i <= max; i++) { bild[i] = name + i + '.jpg'; }
   var Fenster;
   window.addEventListener( "unload", function() {Fenster.close();} );

   function PopUp(pict) {
     if (Fenster) { Fenster.close(); }
     Fenster = window.open("","","top=20,left=20,width=900 height=700,
                          resizable=yes,scrollbars=yes");
     var b = "<IMG SRC = " + pict + " height=90%>";
     with (Fenster.document) {
        writeln("<HTML><HEAD><TITLE></TITLE></HEAD>");
        writeln("<BODY style='background-color: black;'>");
        writeln("<br>");
        writeln(b);
        writeln("<br>");
        writeln("</BODY></HTML>");
     }
   }
</script>
</head>

<body id="bo">
<table class="tab">
  <tr><td><a href="#bottom"> nach unten </a></td></tr>
</table>
<p>Bildergalerie - Bild mit Mausklick auswählen</p>

<script>
   var vbody = document.getElementById("bo");
   var vtable = document.createElement("TABLE");
   vbody.appendChild(vtable);

   for (var j = 1; j <= max; j++) {
     var pix = bild[j];
     if ( (j % 5) == 1 ) {
        var vtr = document.createElement("TR");
        vtable.appendChild(vtr);
     }
     var vtd = document.createElement("TD");
     vtr.appendChild(vtd);
     var vimg = document.createElement("IMG");
     vimg.setAttribute("id","img" + j);
     vimg.setAttribute("src",pix);
     vimg.setAttribute("width",200);
     vimg.addEventListener("click", function() { PopUp(this.src); });
     vtd.appendChild(vimg);
     document.getElementById("img" + j).style.border='2px solid silver';
   }
</script>

<br>
<table class="tab">
  <tr><td><a href="#top"> nach oben </a></td></tr>
</table>
<a name = "bottom"></a>
</body>
</html>
```

[2.3.2] Zeitgesteuerte Bildershow

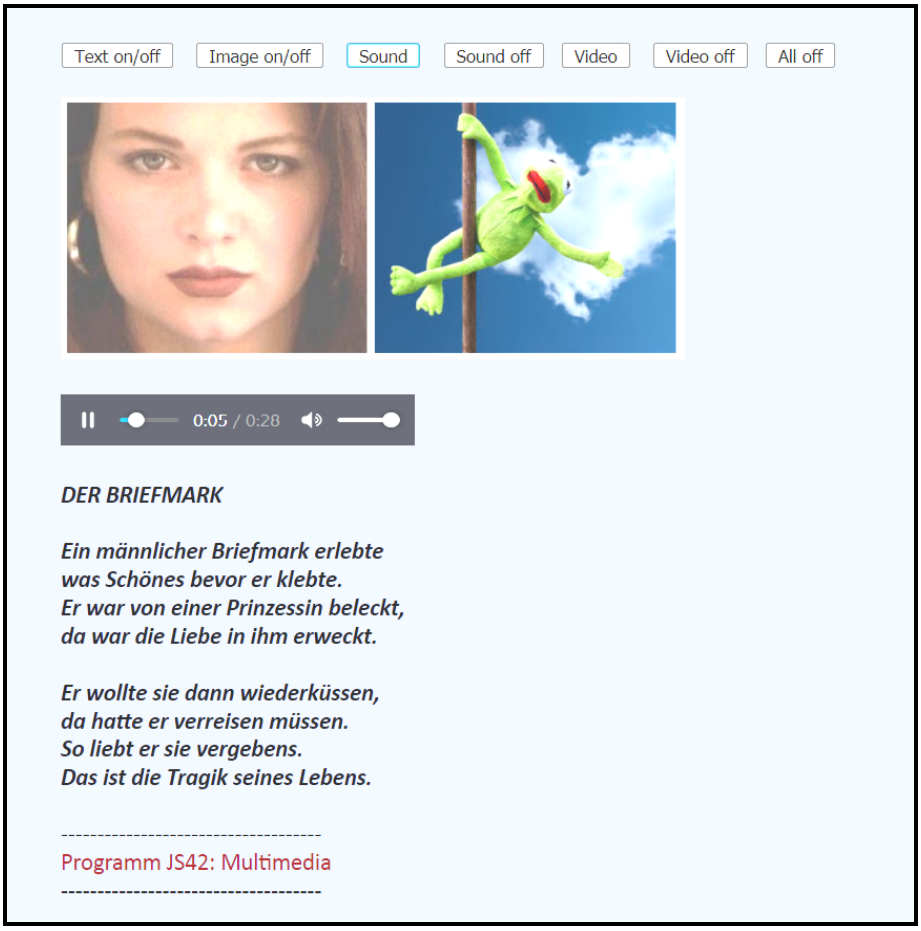
```
<!DOCTYPE html>
<head>
<title>js41.html</title>
<meta charset="ISO-8859-1">
<meta name="author" content="Herbert Paukert">
<style type="text/css">
   body { background-color:#000; color:yellow;
          font-family:'Calibri,sans-serif'; font-size:16px; font-weight:normal;
          text-align:center; margin-left:5%;}
   .btn { background-color: #DDD; border: 2px solid yellow; border-radius:10px; font-size: 18px;}
</style>
```

```
<script>
   var active;               // Hilfsvariable für Bildershow
   var ShowRun = false;      // Kennvariable für Bildershow
   var zeit = 3000;          // Bilder-Darbietungszeit für Bildershow
   var tSound = true;        // Wechselschalter für Hintergrundmusik
   var pictA = new Image();  // Bildspeicher
   var num = 1;              // Bildnummer
   var max = 8;              // Anzahl der Bilder
   var name = 'bild';        // Bildnamen
   bild = new Array(max);    // Array der Bilder

   for (var i = 1; i <= max; i++) { bild[i] = name + i + '.jpg'; }

   window.onload = function() {
     document.formu.ausgabe.value = bild[1];
     pictA.src = bild[1];
     document.pic1.src = pictA.src;
     window.scrollTo(0,0);
    }

   function GotoImage()  {
     if (ShowRun ) { return; }
     zahl = prompt('Nummer der Bildseite zwischen 1 und '+(max),num);;
     if (!isNaN(zahl)) {
       num = zahl - 1;
       if (num <= 0) { num = 0; }
       if (num >= max) { num = max-1 ; }
     }
     else { num = 0; }
     GotoNext();
   }

   function GotoNext() {
     num = num + 1;
     if (num > max) { num = 1; }
     document.formu.ausgabe.value = num + ' / ' + bild[num];
     pictA.src = bild[num];
     document.pic1.src = pictA.src;
     if (ShowRun) { active = window.setTimeout("GotoNext()",zeit);  }
   }

   function GotoLast() {
     if (ShowRun ) { return; }
     num = num - 1;
     if (num < 1) { num = max; }
     document.formu.ausgabe.value = num + ' / ' + bild[num];
     pictA.src = bild[num];
     document.pic1.src = pictA.src;
   }

   function ShowStart() {
     if (ShowRun) { return; }
     ShowRun = true;
     active = window.setTimeout("GotoNext()",zeit);
    }

   function ShowStop() {
     ShowRun = false;
     window.clearTimeOut(active);
   }

   function GetTime() {
     if (ShowRun) { return; }
     zahl = prompt('Darbietungszeit zwischen 100 und 9000 MSec',zeit);
     if (!isNaN(zahl)) {
       if ((zahl < 100) || (zahl > 9000)) { zahl = 3000; }
     }
     else { zahl = 3000; }
     zeit = zahl;
   }

  function SoundToggle() {
     tSound = !tSound;
     if (tSound == true)  { backplay.play();  }
     if (tSound == false) { backplay.pause(); }
  }

</script>
</head>

<body>
<form name = "formu">
<h2> Zeitgesteuerte Bildershow</h2>
<input type = "button" class = "btn" name = "btn1" value = " Goto  " onclick = "GotoImage()"> 
<input type = "button" class = "btn" name = "btn2" value = " Next  " onclick = "GotoNext()"> 
<input type = "button" class = "btn" name = "btn3" value = " Back  " onclick = "GotoLast()"> 
<input type = "button" class = "btn" name = "btn4" value = " Start " onclick = "ShowStart()"> 
<input type = "button" class = "btn" name = "btn5" value = " Stop  " onclick = "ShowStop()"> 
<input type = "button" class = "btn" name = "btn6" value = " Time  " onclick = "GetTime()"> 
<input type = "button" class = "btn" name = "btn7" value = " Sound " onclick = "SoundToggle()"> 
<br><br>
<img src="" id="pic1" name="pic1" style="height:480px; border:2px solid silver; border-radius:10px;">
<br>
<input type = "text" name="ausgabe" value="" size="10" readonly>
<audio id="backplay" class = "btn" loop autoplay> <source src="backsound.mp3" type="audio/mp3"> </audio>
</form>
</body>
</html>
```

[2.3.3] Multimedia

Im folgenden Programm werden ein Audio-Objekt und ein Video-Objekt definiert und mit Hilfe von JavaSript über entsprechende Schalter gesteuert. Die zwei wichtigsten Schalterfunktionen sind *play()* und *stop()*. Außerdem können alle visuellen Objekte eingeblendet oder ausgeblendet werden.

```
<audio id="splay" width="320" controls="true">
  <source src="Audio-Name" type="audio/mp3">
</audio>

<video id="vplay" width="640" controls="true">
  <source src="Video-Name " type="video/mp4">
</video>
```

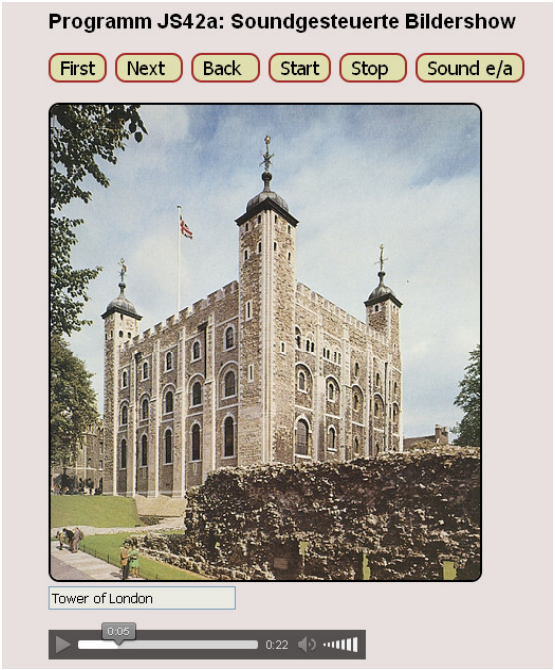



```
<!DOCTYPE html>
<html>
<head>
<title> JS42 </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" ">
<meta name="keywords" content=" ">

<style type="text/css">
    body{background-color:#C0C8CF; color:black;
    font-family:Calibri; font-size:18px; font-weight:normal; text-size-adjust: none;
    text-align:left; margin-left:5%;}
    div{font-weight: bold; font-style: italic;}
</style>
```

```
<script>
function Refresh() { window.location.reload(); };

function twohide() {
   document.getElementById("pict").style.visibility = "hidden";
   document.getElementById("poem").style.visibility = "hidden";
}
videoplay = function(ein) {
   document.getElementById('vplay').src = ein; vplay.width="640"; vplay.controls="true";
   vplay.play(); vplay.style.visibility = "visible";
}
soundplay = function(ein) {
   document.getElementById('splay').src = ein; splay.width="320"; splay.controls="true";
   splay.play(); splay.style.visibility = "visible";
}
soundoff = function() {
   document.getElementById('splay').pause(); splay.style.visibility = "hidden";
}
videooff = function() {
   document.getElementById('vplay').pause(); vplay.style.visibility = "hidden"
}
var sichtbar1 = false;
function imageshow() {
   var e1 = document.getElementById("pict");
   sichtbar1 = !sichtbar1;
   if (sichtbar1) { e1.style.visibility = "visible"; }
   else { e1.style.visibility = "hidden"; }
}
var sichtbar2 = false;
function textshow() {
   var e2 = document.getElementById("poem");
   sichtbar2 = !sichtbar2;
   if (sichtbar2) { e2.style.visibility = "visible"; }
   else { e2.style.visibility = "hidden"; }
}
</script>
</head>

<body onload="twohide();">
<br>

<video id="vplay" width="0">
  <source src=" " type="video/mp4">
</video>

<br><br>
<input type="button" value=" Text on/off " onClick="textshow();"/>   
<input type="button" value=" Image on/off " onClick="imageshow();"/>   
<input type="button" value=" Sound " onClick="soundplay('briefmark.mp3');"/>   
<input type="button" value=" Sound off " onClick="soundoff();"/>  
<input type="button" value=" Video " onClick="videoplay('schi.mp4');"/>   
<input type="button" value=" Video off " onClick="videooff();"/>  
<input type="button" value=" All off " onClick="Refresh();"/>  
<br><br>

<img id="pict" src="fraufrog.jpg" height="200" alt="bild">
<br><br>

<audio id="splay" width="0">
  <source src=" " type="audio/mp3">
</audio>

<div id="poem">
<br>
   DER BRIEFMARK<br>
<br>
   Ein männlicher Briefmark erlebte<br>
   was Schönes bevor er klebte.<br>
   Er war von einer Prinzessin beleckt,<br>
   da war die Liebe in ihm erweckt.<br>
<br>
   Er wollte sie dann wiederküssen,<br>
   da hatte er verreisen müssen.<br>
   So liebt er sie vergebens.<br>
   Das ist die Tragik seines Lebens.<br>
</div>
</body>
</html>
```

[2.3.4] Soundgesteuerte Bildershow

Im folgenden Programm sind JPEG-Bilder mit MP3-Sounds verknüpft (synchronisiert), indem sie denselben Dateinamen haben, z.B. „london1.jpg“ und „london1.mp3“.
Insgesamt liegen neun solche Verknüpfungen vor, deren Namen in zwei getrennten Arrays („bild“ und „ton“) abgespeichert werden. Mit entsprechenden Schaltern (<First>, <Next>, <Back>, <Start>, <Stop>) werden die Bilder angezeigt und die passenden Sounds abgespielt. Mit dem Wechselschalter <Sound e/a> wird der Sound ein- und ausgeschaltet.



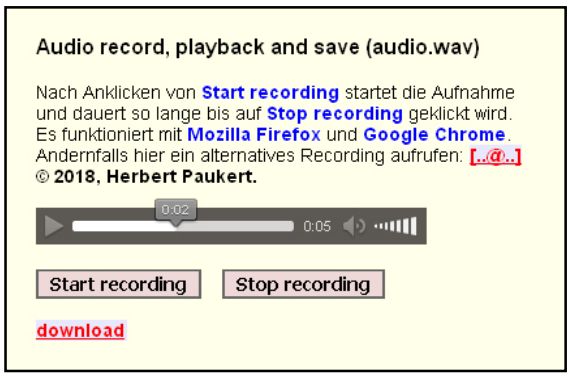

```
<!DOCTYPE html>
<html>
<head>
<title> JS42a </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Herbert Paukert">

<style type="text/css">
   body { background-color:#E8DFDF; color:black;font-family:'Arial'; font-size:16px;
          text-align:left; margin-left:5%;}
   .btn { background-color: #DFDFAF; border: 2px solid brown; border-radius:10px; font-size: 18px;}
</style>

<script>
   var active;                  // Hilfsvariable für Bildershow
   var showrun = false;         // Kennvariable für Bildershow
   var zeit = 3000;             // Bilder-Darbietungszeit für Bildershow OHNE Sound
   var tsound = true;           // Wechselschalter für Sound
   var num = 1;                 // Bildnummer
   var max = 9;                 // Anzahl der Bilder
   var ort = "london";          // Dateinamen der Bilder ('london' + 'num' + '.jpg')
   var bild  = new Array(max);  // Array der Bildnamen
   var ton   = new Array(max);  // Array der Soundnamen
   var titel = new Array(max);  // Array der Bildtitel

   var s1="0,Tower Bridge,Tower of London,Houses of Parliament,Buckingham Palace,Downing Street,";
   var s2="Westminster Abbey,St. Paul's Cathedral,Trafalgar Square,Piccadilly Circus";
   var s = s1 + s2;
   titel = s.split(',');

   for (var i = 1; i <= max; i++) {
      bild[i] = new Image();
      bild[i].src = ort + i + ".jpg";
      ton[i] = ort + i + ".mp3";
   }

   function ShowFirst() {
   // Zum ersten Bild
      num = 1;
      document.formu.animation.src = bild[1].src;
      document.formu.ausgabe.value = titel[1];
      if (tsound) { soundplay(ton[1]) }
      else { splay.pause(); }
   }
```

```
   function ShowNext() {
   // Zum nachfolgenden Bild
      num = num + 1;
      if (num > max) { num = 1; }
      document.formu.animation.src = bild[num].src;
      document.formu.ausgabe.value = titel[num];
      if (tsound) {
         soundplay(ton[num]);
         if (showrun) {splay.onplaying = function() { isSound(); }; }
      }
      if (!tsound) {
         splay.pause();
         if (showrun) {active = setTimeout('ShowNext()', zeit); }
         else { clearTimeout(active) };
      }
   }

   function ShowLast() {
   // Zum vorangehenden Bild
      num = num - 1;
      if (num < 1) { num = max; }
      document.formu.animation.src = bild[num].src;
      document.formu.ausgabe.value = titel[num];
      if (tsound) { soundplay(ton[num]) }
      else { splay.pause(); }
   }

   function ShowStart() {
   // Bildershow starten
      GetTime();
      showrun = true;
      ShowNext();
   }

   function ShowStop() {
   // Bildershow beenden
      alert('Show-Stop');
      showrun = false;
   }

   soundplay = function(ein) {
      document.getElementById('splay').src = ein;
      splay.width ="320"; splay.controls="true"; splay.play();
   }

  function isSound() {
      splay.onended = function() {
        if ( showrun ) { active = setTimeout('ShowNext()', 100); }
        else { clearTimeout(active) };
      }
  }
   function GetTime() {
      if (showrun) { return; }
      zahl = prompt('Show-Darbietungszeit OHNE Sound zwischen 100 und 9000 MSec',zeit);
      if (!isNaN(zahl)) {
        if ((zahl < 100) || (zahl > 9000)) { zahl = 3000; }
      }
      else { zahl = 3000; }
      zeit = zahl;
   }

  function SoundToggle() {
      tsound = !tsound;
      if (tsound) { splay.play();}
      if (!tsound){
        splay.pause();
        if (showrun) {active = setTimeout('ShowNext()', zeit); }
      }
  }
</script>
</head>

<body onload="ShowFirst()">
<form name = "formu">
<h3>Programm JS42a: Soundgesteuerte Bildershow</h3>
<input type = "button" class = "btn" value = " First " onclick = "ShowFirst()"> 
<input type = "button" class = "btn" value = " Next  " onclick = "ShowNext()"> 
<input type = "button" class = "btn" value = " Back  " onclick = "ShowLast()"> 
<input type = "button" class = "btn" value = " Start " onclick = "ShowStart()"> 
<input type = "button" class = "btn" value = " Stop  " onclick = "ShowStop()"> 
<input type = "button" class = "btn" value = " Sound e/a " onclick = "SoundToggle()"> 
<br><br>
<img src=" " name="animation" style="height:450px; border:2px solid black; border-radius:10px;">
<br>
<input type = "text" name="ausgabe" value=" " size="25" readonly>
<br><br>
<audio id="splay" width="320" heigth="240">
  <source src=" " controls="true" type="audio/mp3">
</audio>
</form>
</body>
</html>
```

[2.3.5] Soundrecording mit dem Mikrofon

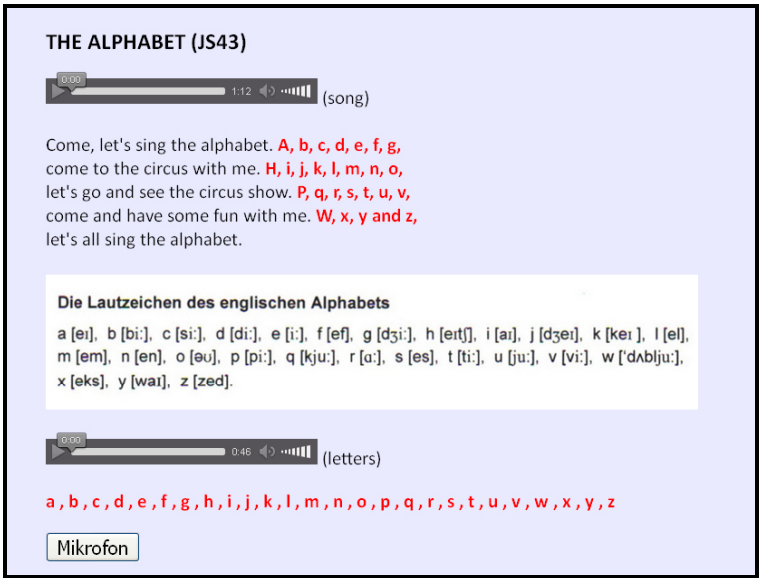
Das folgende Programm „***paumicro.html***“ ist für Mikrofonaufnahmen entwickelt worden. Der Programmcode setzt tiefere technische Kenntnisse voraus. Ausführliche Erklärungen dazu findet man in Teil 7 ab Buchseite 331.

```
<!DOCTYPE html>
<html>
<head>
<title>Mikrofon</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="author" content="Paukert Herbert">

<style>
body { background-color:#FFFFEA; color:black;
       font-family:sans-serif,System; font-size:14px; text-align:left; padding:2% }
.cd { border:2px solid #666666; background-color:#EED8D8; font-size:14px; font-weight: bold; }
a:link { color:red; background-color:#E8E8FF; font-weight:bold; text-decoration:underline; }
</style>

<script>

navigator.mediaDevices.getUserMedia({audio:true}).then(stream => {
   rec = new MediaRecorder(stream);
   audioChunks = [];
   rec.ondataavailable = e => {
      audioChunks.push(e.data);
      if (rec.state == "inactive") {
        let blob = new Blob(audioChunks,{type:'audio/wav'});
        recAudio.src = URL.createObjectURL(blob);
        recAudio.controls = true;
        recAudio.autoplay = true;
        createAudioLink(recAudio.src);
      }
   }
}).catch(e => alert(e));

function createAudioLink(blobUrl) {
    let downloadEl = document.createElement('a');
    downloadEl.style = 'display: block';
    downloadEl.innerHTML = 'download';
    downloadEl.download = 'audio.wav';
    downloadEl.href = blobUrl;
    document.body.appendChild(downloadEl);
}

function startRecord() {
  show();
  startRecord.disabled = true;
  stopRecord.disabled = false;
  audioChunks = [];
  rec.start();
}

function stopRecord() {
  hide();
  startRecord.disabled = false;
  stopRecord.disabled = true;
  rec.stop();
}

function show() { document.getElementById("info").style.visibility = 'visible'; }
function hide() { document.getElementById("info").style.visibility = 'hidden';  }

</script>
</head>

<body onload="hide()">
<form>
<h3>Audio record, playback and save (audio.wav)</h3>
  Nach Anklicken von <b>Start recording</b> startet die Aufnahme<br>
  und dauert so lange bis auf <b>Stop recording</b> geklickt wird.<br>
  Es funktioniert mit <b> Mozilla Firefox</b> und <b>Google Chrome</b>.<br>
  Andernfalls hier ein alternatives Recording aufrufen:
  <a href="https://www.speakpipe.com/voice-recorder">[..@..]</a><br>
  <b> © 2018, Herbert Paukert.</b>
<br><br>
<audio id="recAudio" controls="true">
</audio>
<br><br>
<input type="button" id="start" value="Start recording" size="4" class="cd" onclick= "startRecord()"> 
<input type="button" id="stop" value="Stop recording" class="cd" size="4" onclick= "stopRecord()"><br>
<div id="info">
<i>Bitte jetzt sprechen . . .</i>
</div>
</form>
</body>
</html>
```



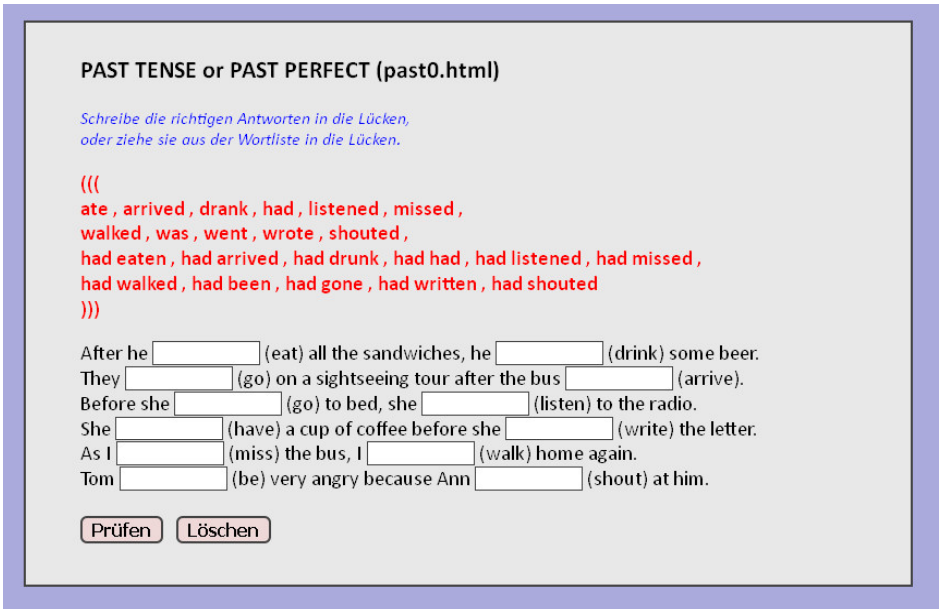
[2.4] Entwicklung von Lernprojekten (Englisch und Mathematik)

[2.4.1] The English alphabet




```
<!doctype html>
<html>
<head>
<title>The Alphabet</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" ">

<style type = "text/css">
body {background-color:#EAEAFF; color:black; font-family:Calibri,Arial;
      font-size:21px; font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
#buch {color:red; font-weight:bold; }
#mic {font-size:21px; }
</style>

<script>
function NewWindow()  {
   var micro = window.open("paumicro.html","micro",
                           "left=800,width=500,height=600,toolbar=1,scrollbars=1");
}
</script>
</head>

<body>
<h3>THE ALPHABET (JS43)</h3>
<audio width="320" height="240" controls>
   <source src="alpha1.mp3" type="audio/mp3">
</audio> (song)
<br><br>
Come, let's sing the alphabet. <b><font color = "red">A, b, c, d, e, f, g,</font></b><br>
come to the circus with me. <b><font color = "red">H, i, j, k, l, m, n, o,</font></b><br>
let's go and see the circus show. <b><font color = "red">P, q, r, s, t, u, v,</font></b><br>
come and have some fun with me. <b><font color = "red">W, x, y and z,</font></b><br>
let's all sing the alphabet.<br>
<br>
<img src="alphabet.jpg" height="150" alt="bild"><br>
<br>
<audio width="320" height="240" controls>
   <source src="alpha2.mp3" type="audio/mp3">
</audio> (letters)
<br>
<p id="buch"> a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z </p>
<input type="button" value=" Mikrofon " id="mic" onClick="NewWindow()">
<br>
</body>
</html>
```

[2.4.2] Past Tense or Past Perfect (Version 0)



Das hier vorliegende Programm „***past0.html***" ist die Basisversion eines Lückentests. Sie kann als Grundgerüst für beliebige Lückentests dienen.

Im Header der HTML-Seite befindet sich die zentrale JavaScript-Funktion „***pruef()***", wo zuerst das Array der richtigen Antworten steht, bei denen randständige Blanks entfernt werden. Dann werden in einer Schleife diese richtigen Antworten mit den vom Anwender eingegebenen Texten verglichen. Alle richtigen Lösungen (Treffer) werden gezählt, mit einem „+" versehen und ausgegeben. Zum Schluss wird die Anzahl der Treffern angezeigt.

Im Body der HTML-Seite wird die Wortliste der richtigen Antworten angezeigt. (Natürlich kann das Programm auch ohne diese Antwortanzeige erstellt werden). Darunter steht dann das Formular mit den verschiedenen Textlücken-Feldern ***<input type="text" size="10" class="cd1"/>*** . In diese Lückenfelder muss der Anwender seine Eingaben schreiben.

Am Ende der Seite befinden sich zwei Schalter. Mit dem ersten Schalter kann die Funktion „***pruef()***" aufgerufen werden. Der zweite Schalter bewirkt einen Reset des Formulars, wobei alle Lückenfelder wieder gelöscht werden.

```
<!DOCTYPE html>
<html>
<head>
<title>Paste Tense (past0.html)</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Englisches Lernprogramm, Basisversion">

<style>
body   {background-color:#AAAADD; color:black; font-family:Calibri,Arial,sans-serif;
        font-size:19px; font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
.cd1   {border: 1px solid #444444; background-color: #FFFFFF; font-size: 15px;}
.cd2   {border: 2px solid #444444; background-color: #EED8D8; font-size: 16px;
        font-weight: bold; border-radius: 6px;}
.p1    {color: blue; font-size: 16px; font-style: italic;}
.p2    {color: red; font-size: 19px; font-weight: bold;}
.list  {font-size: 19px;}
#fs    {width: 750px; background-color: #E8E8E8; padding-left: 50px;
        border: 2px solid #444444; margin: auto;};
</style>
```

```
<script>
function pruef() {
  var num = 0;
  var antwo = new Array(12);

  antwo[0] = "had eaten";
  antwo[1] = "drank";
  antwo[2] = "went";
  antwo[3] = "had arrived";
  antwo[4] = "went";
  antwo[5] = "had listened";
  antwo[6] = "had had";
  antwo[7] = "wrote";
  antwo[8] = "had missed";
  antwo[9] = "walked";
  antwo[10] = "was";
  antwo[11] = "had shouted";

  var leng = antwo.length;
  for (var i = 0; i < leng; i++) { antwo[i] = antwo[i].trim();  }

  var formular = document.myForm;
  var anz = formular.length;

  for (var i = 0; i < anz; i++)
  {
    let t = antwo[i] + "+";
    let f = antwo[i];
    if (formular.elements[i].value.trim() == antwo[i]) { num++; formular.elements[i].value = t; }
    else { formular.elements[i].value = f; };
  }
  alert(num + " von " + anz + " Treffern (+)");
}

function start() { document.myForm.reset(); }
</script>
</head>

<body oncontextmenu="return false">
<!-- Verhinderung der Quellcode-Anzeige durch Sperre des Kontextmenüs -->
<!-- Lückenfeld-Typ: <input type="text" size="10" class="cd1"/> -->
<!-- Lückenfeld-Variable: (???) -->
<fieldset id="fs">
<h3>PAST TENSE or PAST PERFECT (past0.html)</h3>
<p class="p1">
  Schreibe die richtigen Antworten in die Lücken,<br>
  oder ziehe sie aus der Wortliste in die Lücken.<br>
</p>
<p class="p2">
(((<br>
ate , arrived , drank , had , listened , missed ,<br>
walked , was , went , wrote , shouted ,<br>
had eaten , had arrived , had drunk , had had , had listened , had missed ,<br>
had walked , had been , had gone , had written , had shouted <br>
)))
</p>
<form name="myForm">
<div class=list">
After he <input type="text" size="10" class="cd1"/> (eat) all the sandwiches,
he <input type="text" size="10" class="cd1"/> (drink) some beer.<br>
They <input type="text" size="10" class="cd1"/> (go) on a sightseeing tour
after the bus <input type="text" size="10" class="cd1"/> (arrive).<br>
Before she <input type="text" size="10" class="cd1"/> (go) to bed,
she <input type="text" size="10" class="cd1"/> (listen) to the radio.<br>
She <input type="text" size="10" class="cd1"/> (have) a cup of coffee
before she <input type="text" size="10" class="cd1"/> (write) the letter.<br>
As I <input type="text" size="10" class="cd1"/> (miss) the bus,
I <input type="text" size="10" class="cd1"/> (walk) home again.<br>
Tom <input type="text" size="10" class="cd1"/> (be) very angry
because Ann <input type="text" size="10" class="cd1"/> (shout) at him.<br>
</div>
</form>
<br>
<input type = "button" value=" Prüfen "  onclick="pruef()" class="cd2"/>  
<input type = "button" value=" Löschen " onclick="start()"; class="cd2"/><br>
<br>
<script>  document.forms[0].elements[0].focus(); </script>
</fieldset>
</body>
</html>
```

Hinweis: Bei der Erstellung des Programms müssen nur zwei Abschnitte individuell gestaltet werden: (1) Die Liste der richtigen Antworten und (2) der eigentliche Lückentext. Dabei können alle Eingabefelder im Formular zunächst durch die Zeichenfolge **(???)** im Programmcode niedergeschrieben werden.

```
<form name="myForm">
<div class="list">
After he (???) (eat) all the sandwiches, he (???) (drink) some beer.<br>
They (???) (go) on a sightseeing tour after the bus (???) (arrive).<br>
Before she (???) (go) to bed, she (???) (listen) to the radio.<br>
She (???) (have) a cup of coffee before she  (???) (write) the letter.<br>
As I (???) (miss) the bus, I (???) (walk) home again.<br>
Tom (???) (be) very angry because Ann (???) (shout) at him.<br>
</div>
</form>
```

Dann können im Texteditor die **(???)** durch **<input type="text" size="10" class="cd1"/>** automatisch ersetzt werden. Dadurch wird eine ökonomische Erzeugung von Lückentexten ermöglicht.

[2.4.3] Past Tense or Past Perfect (Version 1)

Das vorliegende Programm „***past.html***" ist eine erweiterte Version des Lückentests „***past0.html***" mit folgenden Erweiterungen:

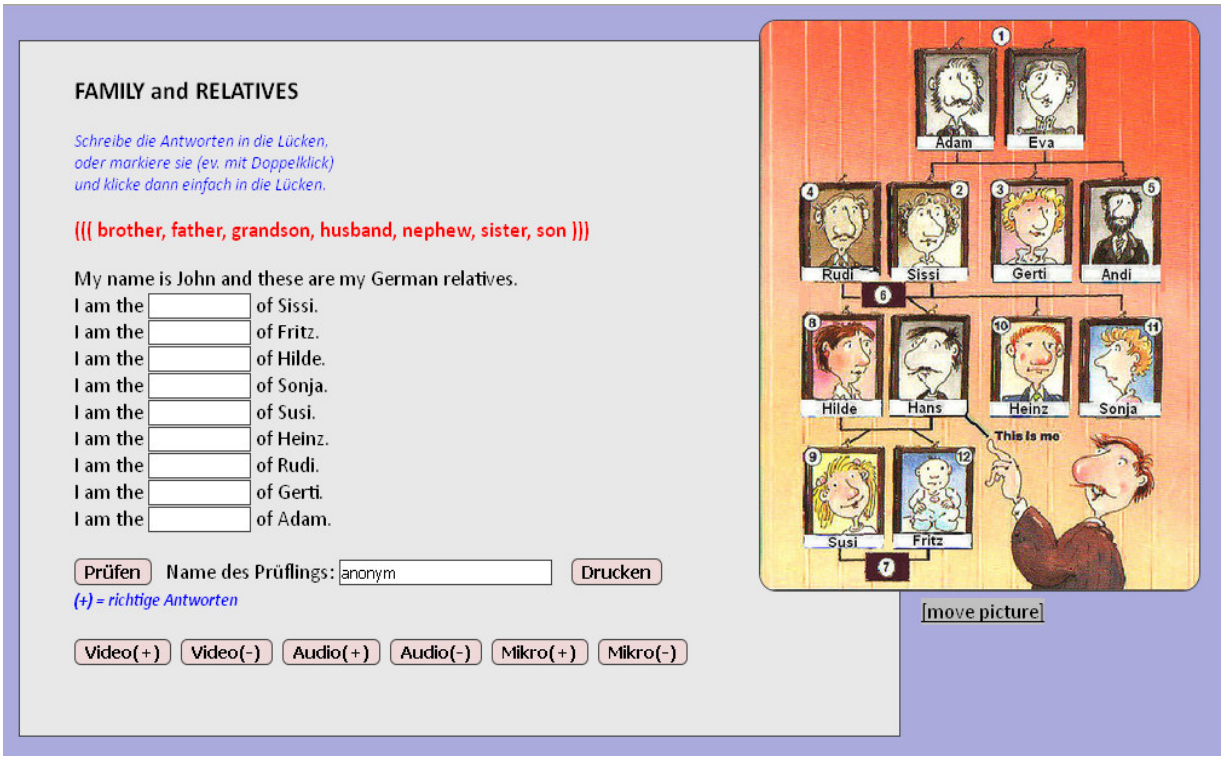
(1) Kopieren der Antworten mittels „Klick-Klick-Technik" statt „Ziehen" mit der Maus. (Zuerst ein Doppelklick auf den HTML-Text und dann ein Einfachklick auf das Zielfeld.)
(2) Alle Texte sind unabhängig von Groß- oder Kleinschrift (case-insensitive). Das kann mit der Steuervariablen „**lcase**" bestimmt werden.
(3) In einem zusätzlichen Textfeld kann der Name des Anwenders eingegeben werden.
(4) Bei der Ergebnisausgabe werden auch das aktuelle Datum und die Uhrzeit angezeigt.
(5) Das gesamte Formular kann gedruckt werden.
(6) Eine Wiederholung des Lückentests ist nicht möglich. Das bewirkt die Steuervariable „**stop**".

```
<!DOCTYPE html>
<html>
<head>
<title>Lückentest</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" ">
<meta name="description" content="Englisches Lernprogramm, Erweiterte Version">
```

```
<style>
body   {background-color:#AAAADD; color:black; font-family:Calibri,Arial,sans-serif;
        font-size:19px; font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
.cd1   {border: 1px solid #444444; background-color: #FFFFFF; font-size: 15px;}
.cd2   {border: 2px solid #444444; background-color: #EED8D8; font-size: 16px;
        font-weight: bold; border-radius: 6px;}
.p1    {color: blue; font-size: 16px; font-style: italic;}
.p2    {color: red; font-size: 19px; font-weight: bold;}
.list  {font-size: 19px;}
#msg   {color: blue; font-size: 16px; font-style: italic; font-weight: bold;}
#myImg {border: 2px solid #444444; border-radius: 16px; }
#fs    {width: 750px; background-color: #E8E8E8; padding-left: 50px;
        border: 2px solid #444444; margin: auto;};
</style>

<script>
var mText = '';
document.onclick = function(event){
// Bestimmt ein Event-Objekt und daraus einen angeklickten Text-Bereich,
// der durch Leerzeichen oder Beistriche begrenzt ist.
// Dieser kann dann mit Mausklick in ein Formularfeld eingefügt werden.
  var obj = event.target;
  let ide = obj.id;
  if ((ide == "id0") && (window.getSelection()) ) {
    mText = myTrim(window.getSelection().toString());
  }
  else {
    if (mText != '') {
       let ind = getIndex(obj);
       document.myForm.elements[ind].value = mText;
    }
  }
}
function getIndex(elem) {
// Ermittelt den Index von Formularfeldern
  for (var i=0; i < document.myForm.elements.length; i++) {
      if (elem == document.myForm.elements[i]) { return i; }
  }
  return -1;
}
function returnKey(ev) {
// Ermöglicht mit der Return-Taste zum nächsten Eingabefeld zu springen
   var taste = ev.which || ev.keyCode;
   if (taste == 13)  {
      let field = ev.target;
      let i = getIndex(field);
      document.myForm.elements[i+1].focus();
   }
}
function Reset()      { window.location.reload(); }             // Programm neu starten
function Refresh()    { document.myForm.reset(); }              // Formular neu initialisieren
function myTrim(x)    { return x.replace(/^\s+|\s+$/gm,''); }   // String trimmen

var lcase = true;     // Unabhängigkeit von Groß-/Kleinschrift
var stop  = false;    // Kennvariable für Testende

function pruef() {
// Die Prüfroutine liefert die richtigen Antworten
// und vergleicht sie mit den Formulareingaben
// und verhindert auch einen wiederholten Aufruf.
  if (stop) { alert('Wiederholungen sind NICHT möglich!'); return; }
  stop = true;
  var num = 0;
  var plus = "+"
  var formular = document.myForm;
  var anz = formular.length;
  var antwo = new Array(anz);

  antwo[0] = "had eaten";
  antwo[1] = "drank";
  antwo[2] = "went";
  antwo[3] = "had arrived";
  antwo[4] = "went";
  antwo[5] = "had listened";
  antwo[6] = "had had";
  antwo[7] = "wrote";
  antwo[8] = "had missed";
  antwo[9] = "walked";
  antwo[10] = "was";
  antwo[11] = "had shouted";
```

```
  for (var i = 0; i < anz; i++) {
    antwo[i] = myTrim(antwo[i]);
    if (lcase) { antwo[i] = antwo[i].toLowerCase(); }
  }

  var s = '';
  for (var i = 0; i < anz; i++) {
    s = antwo[i];
    if (lcase) { formular.elements[i].value = formular.elements[i].value.toLowerCase(); }
    if ( myTrim(formular.elements[i].value) == antwo[i] ) {
         num++; formular.elements[i].value = s + plus;
    }
    else { formular.elements[i].value = s; };
  }
  alert(num + " von " + anz + " Treffern (+)");

  var datum = new Date(); datum = datum.toString(); datum = datum.substring(4,24);
  document.getElementById("msg").innerHTML =
    "(+) = richtige Antworten = " + num + " von " + anz + " (" + datum + ")";

  if (stop) {
    for (var i = 0; i < anz; i++) { document.forms["myForm"].elements[i].readOnly = true; }
  }
}
</script>
</head>

<body onkeydown = "returnKey(event)" onload="Refresh()" oncontextmenu="return false">
<!-- Verhinderung der Quellcode-Anzeige durch Sperre des Kontextmenüs -->
<!-- Lückenfeld-Typ: <input type="text" size="10" class="cd1"/> -->
<!-- Lückenfeld-Variable: (???) -->

<fieldset id="fs">
<h3>PAST TENSE or PAST PERFECT</h3>
<p class="p1">
Schreibe die Antworten in die Lücken,<br>
oder markiere sie (ev. mit Doppelklick)<br>
und klicke dann einfach in die Lücken.
</p>

<p id="id0" class="p2">
((( ate, arrived, drank, had, listened, missed,<br>
walked, was, went, wrote, shouted,<br>
had eaten, had arrived, had drunk, had had, had listened, had missed,<br>
had walked, had been, had gone, had written, had shouted )))
</p>

<form name="myForm">
<div class="list">
After he  <input type="text" size="10" class="cd1"/> (eat) all the sandwiches, he  <input type="text"
size="10" class="cd1"/> (drink) some beer.<br>
They  <input type="text" size="10" class="cd1"/> (go) on a sightseeing tour after the bus  <input
type="text" size="10" class="cd1"/> (arrive).<br>
Before she  <input type="text" size="10" class="cd1"/> (go) to bed, she  <input type="text" size="10"
class="cd1"/> (listen) to the radio.<br>
She  <input type="text" size="10" class="cd1"/> (have) a cup of coffee before she  <input type="text"
size="10" class="cd1"/> (write) the letter.<br>
As I  <input type="text" size="10" class="cd1"/> (miss) the bus, I  <input type="text" size="10"
class="cd1"/> (walk) home again.<br>
Tom  <input type="text" size="10" class="cd1"/> (be) very angry because Ann  <input type="text"
size="10" class="cd1"/> (shout) at him.<br>
</div>
</form>

<br>
<input type = "button" value=" Prüfen " onclick="pruef()" class="cd2"/>  
Name des Prüflings: <input type = "text" size="25" value="anonym" class="cd1"/>   
<input type="button" value=" Drucken " onclick="print()" class="cd2"/> <br>
<span id = "msg">(+) = richtige Antworten</span><br>
<br>

<script> document.forms[0].elements[0].focus(); </script>
</fieldset>
</body>
</html>
```

[2.4.4] Family and Relatives „*relatives.html*“



Das vorliegende Programm „***relatives.html***” baut auf dem Lückentest „***past.html***” auf - mit folgenden drei multimedialen Erweiterungen.

(1) Darstellung eines Images, das mit der Maus beliebig positioniert werden kann (z.B. family.jpg).
(2) Ein Soundplayer und ein Videoplayer können eingeblendet werden (z.B. family.mp3, family.mp4).
(3) Zusätzlich sind auch Mikrofonaufnahmen zum Sprachtraining möglich (paumicro.html).

Mit diesem Programm wird eine optimale Struktur von multimedialen Lückentests vorgegeben. Damit können dann beliebige Testprogramme einfach und schnell erzeugt werden, indem im Programmcode nur die Antwortliste und der eigentliche Lückentext abgeändert werden. Optional können auch jeweils ein Bildname, Soundname und Videoname eingegeben werden.

```
<!DOCTYPE html>
<html>
<head>
<title>Lückentest</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" ">
<meta name="keywords" content=" ">

<style>
body   {background-color:#AAAADD; color:black; font-family:Calibri,Arial,sans-serif;
        font-size:19px; font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
.cd1   {border: 1px solid #444444; background-color: #FFFFFF; font-size: 15px;}
.cd2   {border: 2px solid #444444; background-color: #EED8D8; font-size: 16px;
        font-weight: bold; border-radius: 6px;}
.p1    {color: blue; font-size: 16px; font-style: italic;}
.p2    {color: red; font-size: 19px; font-weight: bold;}
.list  {font-size: 19px;}
#msg   {color: blue; font-size: 16px; font-style: italic; font-weight: bold;}
#myImg {border: 2px solid #444444; border-radius: 16px; }
#fs    {width: 750px; background-color: #E8E8E8; padding-left: 50px;
        border: 2px solid #444444; margin: auto;};
</style>
```

```
<script>

var mText = '';

document.onclick = function(event){
// Bestimmt ein Event-Objekt und daraus einen angeklickten Text-Bereich,
// der durch Leerzeichen oder Beistriche begrenzt ist.
// Dieser kann dann mit Mausklick in ein Formularfeld eingefügt werden.
  var obj = event.target;
  let ide = obj.id;
  if ((ide == "id0") && (window.getSelection()) ) {
    mText = myTrim(window.getSelection().toString());
  }
  else {
    if (mText != '') {
       let ind = getIndex(obj);
       document.myForm.elements[ind].value = mText;
    }
  }
}

function Reset()      { window.location.reload(); }     // Programm neu starten
function Refresh()    { document.myForm.reset(); }      // Formular neu initialisieren

// String trimmen
function myTrim(x)  { return x.replace(/^\s+|\s+$/gm,''); }

// Erster Zusatz: Mikrofon mittels "paumicro.html"

var micro;
function NewWindow() {
   micro =
   window.open("paumicro.html","micro","left=10,top=10,width=400,height=470,toolbar=1,scrollbars=1");
   soundstop(); videostop(); }
function ShutWindow() { micro.close(); }

// Zweiter Zusatz: Soundplayer und Videoplayer einblenden

var ein;
soundplay = function(ein) { document.getElementById('splay').src = ein; splay.width="320";
  splay.controls="true"; splay.play(); document.getElementById('splay').style.visibility = "visible";
  videostop(); ShutWindow();
}
soundstop = function(ein) { document.getElementById('splay').src = ein; splay.width="0";
  splay.controls="false"; splay.pause(); document.getElementById('splay').style.visibility = "hidden"
}
videoplay = function(ein) { document.getElementById('vplay').src = ein; vplay.width="320";
  vplay.controls="true"; vplay.play(); soundstop(); ShutWindow(); }
videostop = function(ein) { document.getElementById('vplay').src = ein; vplay.width="0";
  vplay.controls="false"; vplay.pause(); }

// Dritter Zusatz: Objektpositionierung mit der Maus (z.B. für Images)

function dragElement(elem) {
// Verschieben von "div"-Objekten mit der Maus
// Hauptprogramm mit drei Unterprogrammen
   var pos1 = 0, pos2 = 0, pos3 = 0, pos4 = 0;
   elem.onmousedown = dragMouseDown;

   function dragMouseDown(ev) {
   // Erstes Unterprogramm
      ev.preventDefault();
      pos3 = ev.clientX;                              // aktuelle Mausposition X
      pos4 = ev.clientY;                              // aktuelle Mausposition Y
      document.onmousemove = elementDrag;
      document.onmouseup = closeDragElement;
   }

   function elementDrag(ev) {
   // Zweites Unterprogramm
      ev.preventDefault();
      pos1 = pos3 - ev.clientX;
      pos2 = pos4 - ev.clientY;
      pos3 = ev.clientX;
      pos4 = ev.clientY;
      elem.style.top = (elem.offsetTop - pos2) + "px";
      elem.style.left = (elem.offsetLeft - pos1) + "px";
   }

   function closeDragElement() {
   // Drittes Unterprogramm
      document.onmousemove = null;
      document.onmouseup = null;
   }
}
```

```
function getIndex(elem) {
// Ermittelt den Index von Formularfeldern
  for (var i=0; i < document.myForm.elements.length; i++) {
      if (elem == document.myForm.elements[i]) { return i; }
  }
  return -1;
}

function returnKey(ev) {
// Ermöglicht mit der Return-Taste zum nächsten Eingabefeld zu springen
   var taste = ev.which || ev.keyCode;
   if (taste == 13)  {
      let field = ev.target;
      let i = getIndex(field);
      document.myForm.elements[i+1].focus();
   }
}

var lcase = false;     // Unabhängigkeit von Groß-/Kleinschrift
var stop  = false;     // Kennvariable für Testende

function pruef() {
// Die Prüfroutine liefert die richtigen Antworten
// und vergleicht sie mit den Formulareingaben
// und verhindert auch einen wiederholten Aufruf
  if (stop) { alert('Wiederholungen sind NICHT möglich!'); return; }
  stop = true;
  var num = 0;
  var plus = "+"
  var formular = document.myForm;
  var anz = formular.length;
  var antwo = new Array(anz);

// Antwortliste – Anfang
  antwo[0] = "son";
  antwo[1] = "father";
  antwo[2] = "husband";
  antwo[3] = "brother";
  antwo[4] = "father";
  antwo[5] = "brother";
  antwo[6] = "son";
  antwo[7] = "nephew";
  antwo[8] = "grandson";
// Antwortliste - Ende

  for (var i = 0; i < anz; i++) {
    antwo[i] = myTrim(antwo[i]);
    if (lcase) { antwo[i] = antwo[i].toLowerCase(); }
  }

  var s = '';
  for (var i = 0; i < anz; i++) {
    s = antwo[i];
    if (lcase) { formular.elements[i].value = formular.elements[i].value.toLowerCase(); }
    if ( myTrim(formular.elements[i].value) == antwo[i] ) {
         num++; formular.elements[i].value = s + plus;
    }
    else { formular.elements[i].value = s; };
  }

  alert(num + " von " + anz + " Treffern (+)");
  var datum = new Date(); datum = datum.toString(); datum = datum.substring(4,24);
  document.getElementById("msg").innerHTML =
    "(+) = richtige Antworten = " + num + " von " + anz + " (" + datum + ")";

  if (stop) {
    for (var i = 0; i < anz; i++) { document.forms["myForm"].elements[i].readOnly = true; }
  }
}

</script>
</head>
```

```
<body onkeydown = "returnKey(event)" onload="Refresh()" oncontextmenu="return false">
<!-- Verhinderung der Quellcode-Anzeige mittels Kontextmenü -->
<!-- Lückenfeld-Typ: <input type="text" size="10" class="cd1"/> -->
<!-- Lückenfeld-Variable: (???) -->

<fieldset id="fs">
<h3>FAMILY and RELATIVES</h3>
<p class="p1">
Schreibe die Antworten in die Lücken,<br>
oder markiere sie (ev. mit Doppelklick)<br>
und klicke dann einfach in die Lücken.
</p>

<p id="id0" class="p2">
((( brother, father, grandson, husband, nephew, sister, son )))
</p>

<form name="myForm">
<div class="list">
My name is John and these are my German relatives.<br>
I am the <input type="text" size="10" id="id1" class="cd1"/> of Sissi.<br>
I am the <input type="text" size="10" id="id2" class="cd1"/> of Fritz.<br>
I am the <input type="text" size="10" id="id3" class="cd1"/> of Hilde.<br>
I am the <input type="text" size="10" id="id4" class="cd1"/> of Sonja.<br>
I am the <input type="text" size="10" id="id5" class="cd1"/> of Susi.<br>
I am the <input type="text" size="10" id="id6" class="cd1"/> of Heinz.<br>
I am the <input type="text" size="10" id="id7" class="cd1"/> of Rudi.<br>
I am the <input type="text" size="10" id="id8" class="cd1"/> of Gerti.<br>
I am the <input type="text" size="10" id="id9" class="cd1"/> of Adam.<br>
</div>
</form>

<br><input type = "button" value=" Prüfen " onclick="pruef()" class="cd2"/>  
Name des Prüflings: <input type = "text" size="20" value="anonym" class="cd1"/>   
<input type="button" value=" Drucken " onclick="print()" class="cd2"/> 
<br><span id = "msg">(+) = richtige Antworten</span>
<br><br>
<input type="button" value=" Video(+) " onclick="videoplay('family.mp4')" class="cd2"/> 
<input type="button" value=" Video(-) " onclick="videostop()" class="cd2"/> 
<input type="button" value=" Audio(+) " onclick="soundplay('family.mp3')" class="cd2"/> 
<input type="button" value=" Audio(-) " onclick="soundstop()" class="cd2"/> 
<input type="button" value=" Mikro(+) " onclick="NewWindow()" class="cd2"/> 
<input type="button" value=" Mikro(-) " onclick="ShutWindow()" class="cd2"/> 
<br><br>
<video id="vplay" width="0"> <source src=" " type="video/mp4"> </video>
<audio id="splay" width="0"> <source src=" " type="audio/mp3"> </audio>

<div id="pix" style = "position: absolute; left:900px; top: 10px;">
<img id="myImg" src="family.jpg" height="500"><br>
<span id="info" style = "position: absolute; left:150px; background:silver;">
<u>[move picture]</u>
</span>
</div>

<script>
   // Bild mit gehaltener Maus verschieben
   dragElement(document.getElementById("pix"));
   document.forms[0].elements[0].focus();
</script>

</fieldset>
</body>
</html>
```

Hinweis: So wie in der Basisversion kann der eigentliche Lückentext mit **(???)** als Variable für die Lücken bequem eingegeben werden.

```
I am the (???) of Sissi.
I am the (???) of Fritz.
I am the (???) of Hilde.
I am the (???) of Sonja.
I am the (???) of Susi.
I am the (???) of Heinz.
I am the (???) of Rudi.
I am the (???) of Gerti.
I am the (???) of Adam.
```

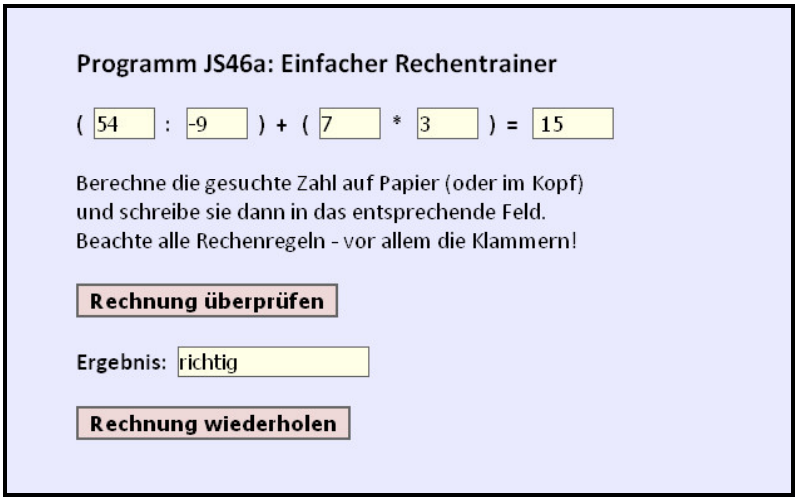
Danach muss nur mehr im Editor **(???)** durch **<input type="text" size="10" class="cd1"/>** ersetzt werden.

[2.4.5] Zahlenraten mit „*raten.html*“


```
<!DOCTYPE html>
<html>
<head>
<title>Zahlen raten</title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Zahlen raten">

<style>
body { background-color:#CCCCEE; color:black; font-family:Arial;
       font-size:18px; font-weight:normal; text-align:left; margin:2%; padding-left:5%; }
#erg { background-color:antiquewhite; color:blue; border:2px solid #666666;
       font-family: Arial; font-size: 20px; font-weight:bold; }
#box { background-color:#EEEEEE; width: 32%; }
.tex { background-color:antiquewhite; border:2px solid #666666; font-size: 16px; font-weight:bold; }
.btn { background-color:antiquewhite; border:2px solid #666666; border-radius:20%;
       font-size: 14px; font-weight:bold; }
</style>

<script>
  var run = false;
  var max = 100;
  var min = 0;
  var text = '';
  var count = 0;
  var zahl;
  var eingabe;
  var ausgabe;

  function fnew() {
    var diff = max - min;
    zahl = Math.floor(diff * Math.random()) + min;
    count = 0;
    text = '';
    document.getElementById("erg").value = text;
    document.getElementById("aus").value = 0;
    document.getElementById("ein").value = '';
    document.getElementById("ein").focus();
    run = true;
  }

  function fplay() {
    if (!run) {
       alert('Zuerst <Neues Spiel> anklicken !');
       return;
    }
    var info1 = ' ist zu hoch !';
    var info2 = ' ist zu tief !';
    var info3 = ' ist richtig !';
    var element = document.getElementById("ein");
    eingabe = element.value.trim();
    count++;
    document.getElementById("aus").value = count;
```

```
    if (eingabe == '' || isNaN(eingabe)) {
       alert('Das ist keine Zahl !');
       element.value = '';
       element.focus();
       return;
    }
    if (zahl < eingabe)  { text = ' ' + eingabe + info1; }
    if (zahl > eingabe)  { text = ' ' + eingabe + info2; }
    if (zahl == eingabe) { text = '  ' + eingabe + info3; run = false; }
    document.getElementById("erg").value = text;
    element.value = '';
    element.focus();
  }
</script>
</head>

<body>
<form name="MyForm">
<br>
<h2>Einfaches Zahlenraten</h2>
<p>
Der Zufall bestimmt eine Zahl zwischen 0 und 100.<br>
Diese soll schnell erraten werden . . .<br>
</p>
<input type="button" name="btn1" value="Neues Spiel" class="btn" onclick="fnew()">  
Zahl eingeben: <input type="text" name="ein" value="0" size="5" id="ein" class="tex">  
<input type="button" name="btn2" value="Zahl testen" class="btn" onclick="fplay()">
<br><br>
<fieldset id="box">
<br>
Ergebnis:  <input type="text" name="ergebnis" value="0" size="18" id="erg" class="tex" readonly>
<br><br>
Eingaben: <input type="text" name="aus" value="0" size="5" id="aus" class="tex" readonly>
<br><br>
</fieldset>
</form>
</body>
</html>
```

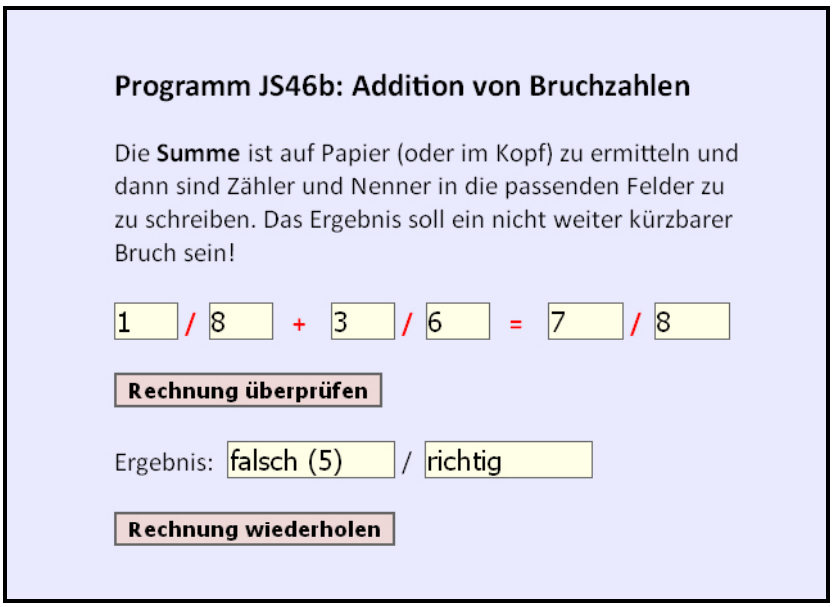
[2.4.6] Rechentraining mit ganzen Zahlen




```
<!DOCTYPE html>
<html>
<head>
<title> JS46a </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Rechentraining mit ganzen Zahlen">

<style>
   body {background-color:#EAEAFF; color:black; font-family:Calibri,Arial;
         font-size:19px; font-weight:normal; text-align:left; margin:2%; padding-left:10%;}
   .cd1 {border: 1px solid #666666; background-color: #FFFFE8; font-size: 17px;}
   .cd2 {border: 2px solid #666666; background-color: #EED8D8; font-size: 17px;
         font-weight: bold;}
</style>
```

```
<script>

function myTrim(x) { return x.replace(/^\s+|\s+$/gm,''); }
function round2(x) { return Math.round(100*x)/100 }

function Pruef() {
   p1 = parseInt(document.MyForm.ein1.value);
   p2 = parseInt (document.MyForm.ein2.value);
   p3 = parseInt (document.MyForm.ein3.value);
   p4 = parseInt (document.MyForm.ein4.value);
   x = (p1 / p2) + (p3 * p4);
   r1 = round2(x);
   a1 = document.MyForm.aus1.value;
   if (myTrim(a1) == r1) { z1 = 'richtig' } else { z1 = 'falsch (' + r1 + ')' };
   document.MyForm.erg1.value = z1;
   document.MyForm.rech.focus();
}
function Neu() {
   document.MyForm.reset();

   z = Math.random(); if (z < 0.5) { vz = -1; } else { vz = 1; }
   z = 1 + 9 * Math.random();
   x = vz * Math.round(z);

   z = Math.random(); if (z < 0.5) { vz = -1; } else { vz = 1; }
   z = 1 + 9 * Math.random();
   y = vz * Math.round(z);

   p1 = Math.round(x * y);
   p2 = Math.round(y);

   z = Math.random(); if (z < 0.5) { vz = -1; } else { vz = 1; }
   z = 1 + 9 * Math.random();
   p3 = vz * Math.round(z);

   z = Math.random(); if (z < 0.5) { vz = -1; } else { vz = 1; }
   z = 1 + 9 * Math.random();
   p4 = vz * Math.round(z);

   document.MyForm.ein1.value = p1;
   document.MyForm.ein2.value = p2;
   document.MyForm.ein3.value = p3;
   document.MyForm.ein4.value = p4;
   document.MyForm.aus1.value = '';
   document.MyForm.erg1.value = '';
   document.MyForm.aus1.focus();
}

</script>
</head>

<body>
<form name="MyForm">
<h3>Programm JS46a: Einfacher Rechentrainer</h3>
<b>
(  <input type="text" name="ein1" value="54" size="2" class="cd1">  :  
<input type="text" name="ein2" value="-9" size="2" class="cd1">  )  +  
(  <input type="text" name="ein3" value="7" size="2" class="cd1">  *  
<input type="text" name="ein4" value="3" size="2" class="cd1">  )  =  
<input type="text" name="aus1" value=" " size="4" class="cd1"><br>
</b>
<br>

Berechne die gesuchte Zahl auf Papier (oder im Kopf)<br>
und schreibe sie dann in das entsprechende Feld.<br>
Beachte alle Rechenregeln - vor allem die Klammern!<br>
<br>

<input type="button" name="rech" value="Rechnung überprüfen" class="cd2" onclick="Pruef()">
<br><br>
<b>Ergebnis:</b>  <input type=text" name="erg1" value="" size="15" class="cd1"
                        readonly="true">
<br><br>
<input type="button" name="knopf" value="Rechnung wiederholen" class="cd2" onclick=Neu()>
</form>

<script>
  document.MyForm.aus1.focus();
</script>

</body>
</html>
```

[2.4.7] Addition von Bruchzahlen




```
<!DOCTYPE html>
<html>
<head>
<title> JS46b </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Bruchrechnen ">

<style>
body {background-color:#EAEAFF; color:black; font-family:Calibri,Arial;
      font-size:21px; font-weight:normal; text-align:left; margin:2%; padding-left:10%;}
.cd1 {border: 1px solid #666666; background-color: #FFFFE8; font-size: 21px;}
.cd2 {border: 2px solid #666666; background-color: #EED8D8; font-size: 17px; font-weight: bold;}
</style>

<script>

function myTrim(x) { return x.replace(/^\s+|\s+$/gm,''); }
function round2(x) { return Math.round(100*x)/100 }

var p1, p2, p3, p4;  // Aufgaben-Parameter
var a1, a2;          // Antwort-Werte
var r1, r2;          // Lösungen
var z1, z2;          // Ausgabe-Werte

function ggt(a,b)
{
  x = a;
  y = b;
  do
  {
    r = x % y;
    x = y;
    y = r;
  }
  while (r > 0);
  return x;
}

function kgv(a,b)
{
  x = ggt(a,b);
  y = a * b / x;
  return y;
}
```

```
function Pruef(xyz) {
  p1 = parseInt(document.MyForm.ein1.value);
  p2 = parseInt(document.MyForm.ein2.value);
  p3 = parseInt(document.MyForm.ein3.value);
  p4 = parseInt(document.MyForm.ein4.value);

  a1 = document.MyForm.aus1.value;
  a2 = document.MyForm.aus2.value;

  r2 = kgv(p2,p4);
  x = r2 / p2;
  y = r2 / p4;
  r1 = x*p1 + y*p3;

  t = ggt(r1,r2);
  r1 = Math.round(r1 / t);
  r2 = Math.round(r2 / t);

  if (myTrim(a1) == r1) { z1 = 'richtig' } else { z1 = 'falsch (' + r1 + ')' };
  document.MyForm.erg1.value = z1;
  if (myTrim(a2) == r2) { z2 = 'richtig' } else { z2 = 'falsch (' + r2 + ')' };
  document.MyForm.erg2.value = z2;
  document.MyForm.aus1.focus();
}

function Neu() {
  a = 1 + 19 * Math.random();
  b = 1 + 19 * Math.random();
  c = 1 + 19 * Math.random();
  d = 1 + 19 * Math.random();
  p1 = Math.round(a);
  p2 = Math.round(b);
  p3 = Math.round(c);
  p4 = Math.round(d);

  document.MyForm.ein1.value = p1;
  document.MyForm.ein2.value = p2;
  document.MyForm.ein3.value = p3;
  document.MyForm.ein4.value = p4;
  document.MyForm.aus1.value = '';
  document.MyForm.aus2.value = '';
  document.MyForm.erg1.value = '';
  document.MyForm.erg2.value = '';
  document.MyForm.aus1.focus();
}

</script>
</head>

<body>
<form name="MyForm">
<br>
<h3>Programm JS46b: Addition von Bruchzahlen</h3>
Die <b>Summe</b> ist auf Papier (oder im Kopf) zu ermitteln und<br>
dann sind Zähler und Nenner in die passenden Felder zu<br>
zu schreiben. Das Ergebnis soll ein nicht weiter kürzbarer<br>
Bruch sein!<br>
<br>
<b><font color = "red">
<input type="text" name="ein1" value="1" size="1" class="cd1"> / 
<input type="text" name="ein2" value="8" size="1" class="cd1">   +   
<input type="text" name="ein3" value="3" size="1" class="cd1"> / 
<input type="text" name="ein4" value="6" size="1" class="cd1">   =   
<input type=text" name="aus1" value="" size="2" class="cd1"> / 
<input type=text" name="aus2" value="" size="2" class="cd1">
</font></b>
<br><br>
<input type="button" name="rech" value="Rechnung überprüfen" class="cd2" onclick="Pruef()"> <br>
<br>
Ergebnis:  <input type=text" name="erg1" value="" size="10" class="cd1"
readonly="true"> / 
<input type=text" name="erg2" value="" size="10" class="cd1" readonly="true">
<br><br>
<input type="button" name="knopf" value="Rechnung wiederholen" class="cd2" onclick=Neu()>
</form>
<script>
  document.MyForm.aus1.focus();
</script>
</body>
</html>
```

[2.5] Dynamische Erzeugung von DOM-Objekten

[2.5.1] Removing, Appending and Styling von DOM-Objekten

Um ein neues DOM-Objekt zu erzeugen, wird es als Kind-Objekt an ein Eltern-Objekt angehängt. (appendChild). Vorhandene Objekte können entfernt (removeChild) oder ersetzt (replaceChild) werden.

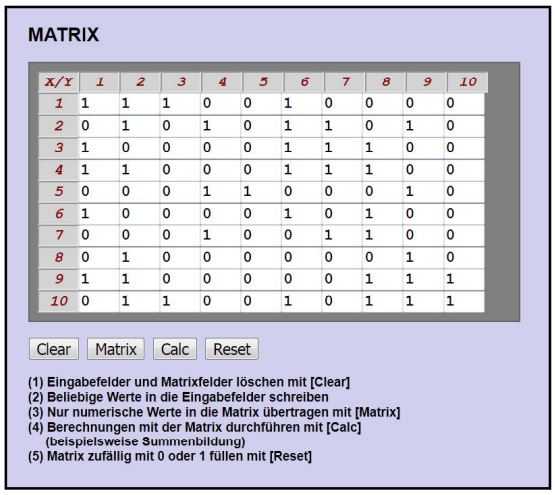
```
<!doctype html>
<html>
<head>
   <title> append.html </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style type="text/css">
       body{background-color:#E0E8EF; color:black;
       font-family:Calibri; font-size:18px; font-weight:normal; text-size-adjust: none;
       text-align:left; margin:5%;}
   </style>
</head>
<body>
   <h3> Removing, Appending and Styling von DOM-Objekten </h3>
   <br>
   <div id = "demo">
      <p id = "p1"> Das ist ein erster Absatz. </p>
      <p id = "p2"> Das ist ein zweiter Absatz. </p>
   </div>
   <script>
      alert('Einen alten Absatz entfernen.');
      var parent = document.getElementById("demo");
      var child = document.getElementById("p1");
      parent.removeChild(child);

      alert('Einen neuen Absatz erzeugen.');
      var abs3 = document.createElement("p");
      abs3.setAttribute("id","p3");
      var txt = document.createTextNode(" Das ist ein neuer Absatz. ");
      abs3.appendChild(txt);
      parent.appendChild(abs3);

      alert('Den Style des Absatzes verändern.');
      document.getElementById("p3").style.color = "red";
      document.getElementById("p3").style.fontStyle = "italic";
      document.getElementById("p3").style.fontWeight = "bold";
      document.getElementById("p3").style.fontSize = "20px";
      document.getElementById("p3").style.textAlign = "center";
   </script>
</body>
</html>
```

- ***Verschiedener Zugriff auf DOM-Objekte***

Beispiel für ein HTML-Element: . . . . .
JavaScript kann auf verschiedene Arten auf ein HTML-Element zugreifen:

- **Über das id-Attribut**: document.getElementById('**inf**');
  (*greift auf das erste gefundene Element zu*)
- **Über den Tag-Namen**: document.getElement**s**ByTagName('**div**');
  (*greift auf alle gefundenen Elemente zu*)
- **Über das class-Attribut**: document.getElement**s**ByClassName('**msg**');
  (*greift auf alle gefundenen Elemente zu*)
- **Über den CSS-Selektor**: (d.h. id, Name oder class)
  document.querySelector('**#inf**');
  (*greift auf das erste gefundene Element zu*)
  document.querySelectorAll('**div**');
  (*greift auf alle gefundenen Elemente zu*)
  document.getElement**s**ByClassName('**msg**');
  (*greift auf alle gefundenen Elemente zu*)

- ***Erzeugung von Matrizen***



Das vorliegende Programm erzeugt in einem div-Element dynamisch Eingabefelder, deren Werte in eine Matrix übertragen werden. Die Matrix ist dabei ein zweidimensionales Array.

[Clear] löscht Eingabefelder und Matrix.
[Matrix] transferiert die Eingaben in die Matrix.
[Calc] führt mit den Feldern der Matrix Berechnungen aus, beispielsweise Zählen aller Felder ungleich Null und Ermittlung deren Summe.
[Reset] füllt die Matrix zufällig mit 0 oder 1

```
<!doctype html>
<html>
<head>
  <title> Matrix </title>
  <meta charset="ISO-8859-1">
  <meta name="description" content="Matrix">
  <meta name="author" content="Herbert Paukert">
  <style>
    body  { background-color:#D0D0EE; color:black;
            font-family:Arial; font-size:15px; font-weight:bold; text-size-adjust: none;
            text-align:left; margin:30px; }
    .btn  { font-size: 18px; }
    #netz { width: 560px; padding: 10px; border:2px solid #666666; background-color: gray; }
  </style>

  <script>
   var max = 10;            // Anzahl
   var mat = new Array();   // Matrix

   document.addEventListener("DOMContentLoaded", function() { init(); })

   function init() {
   // Erzeugung der Eingabefelder und der Matrix
      var parent = document.getElementById("netz");
      for (x = 0; x <= max; x++) {
        for (y = 0; y <= max; y++) {
          var inp = document.createElement("input");
          var s = "i" + x + y;
          inp.setAttribute("id",s);
          inp.setAttribute("size","4");
          inp.setAttribute("value","0");
          parent.appendChild(inp);
          document.getElementById(s).style.fontWeight = "bold";
          document.getElementById(s).style.fontSize = "18px";
          document.getElementById(s).style.fontFamily = "Courier New";
          if (y == 0) {
             inp.setAttribute("value",x);
             inp.setAttribute("readonly",true);
             document.getElementById(s).style.background = "lightgray";
             document.getElementById(s).style.fontStyle = "italic";
             document.getElementById(s).style.color = "darkred";
             document.getElementById(s).style.textAlign = "center";
          }
          if (x == 0) {
             inp.setAttribute("value",y);
             inp.setAttribute("readonly",true);
             document.getElementById(s).style.background = "lightgray";
             document.getElementById(s).style.fontStyle = "italic";
             document.getElementById(s).style.color = "darkred";
             document.getElementById(s).style.textAlign = "center";
          }
        }
        var abs = document.createElement("br");
        parent.appendChild(abs);
      }
      document.getElementById("i00").value = 'X/Y';
      initMatrix();
   }
```

```
   function initMatrix() {
   // die Matrix initialisieren
      for (i = 0; i <= max; i++) { mat[i] = new Array(max); }
      for (i = 1; i <= max; i++) {
        for (j = 1; j <= max; j++) {
          z = Math.floor(Math.random() + 0.5);
          mat[i][j] = z;
          s = "i" + i + j;
          document.getElementById(s).value = z;
        }
      }
   }

   function clearMatrix() {
   // die Matrix und Eingabefelder initialisieren
      for (i = 1; i <= max; i++) {
        for (j = 1; j <= max; j++) {
           mat[i][j] = 0;
           s = 'i' + i + j;
           document.getElementById(s).value = 0;
        }
      }
   }

   function showMatrix() {
   // die Matrix anzeigen
      var s = '';
      for (i = 1; i <= max; i++) {
        s = s + '\n';
        for (j = 1; j <= max; j++) {
           z = mat[i][j];
           s = s + z + ', ';
        }
      }
      alert(s);
   }

   function getMatrix() {
   // die Eingabefelder in eine Matrix transformieren
      for (i = 1; i <= max; i++) {
        for (j = 1; j <= max; j++) {
           x = 'i' + i + j;
           y = document.getElementById(x).value;
           z = parseFloat(y);
           if ( isNaN(z) ) { z = 0; }
           mat[i][j] = z;
        }
      }
      showMatrix();
   }

   function calcMatrix() {
   // Rechnungen mit den Matrixfeldern durchführen
   // z.B. Summe (sum) von allen Feldern <> 0 (count)
      var count = 0, sum = 0, z = 0;
      for (i = 1; i <= max; i++) {
        for (j = 1; j <= max; j++) {
           z = mat[i][j];
           if (z != 0) { count++; sum = sum + z; }
        }
      }
      alert(' Zahlen <> 0:  ' + count + '\n Ihre Summe:  ' + sum);
    }
  </script>
</head>

<body>
<h2>MATRIX</h2>
<div id = "netz"> </div>
<br><br>
<button class="btn" onclick="clearMatrix()"> Clear </button> 
<button class="btn" onclick="getMatrix()"> Matrix </button> 
<button class="btn" onclick="calcMatrix()"> Calc </button>
<button class="btn" onclick="initMatrix()"> Reset </button>
<br><br>
(1) Eingabefelder und Matrixfelder löschen mittels [Clear]<br>
(2) Beliebige Werte in die Eingabefelder schreiben<br>
(3) Nur numerische Werte in die Matrix übertragen mittels [Matrix]<br>
(4) Berechnungen mit der Matrix durchführen mittels [Calc]<br>
     (beispielsweise Summenbildung)<br>
(5) Matrix zufällig mit 0 oder 1 füllen mittels [Calc]<br>
</body>
</html>
```

[2.5.2] Das Spielprogramm „*numbers.html*“

Das Programm demonstriert anschaulich erstens die dynamische Erzeugung von DOM-Objekten und zweitens die Verwendung von CSS-Selektoren mit verschiedenartigen Attributen. Das Listing ist dazu ausführlich kommentiert.


```
<!DOCTYPE html">
<html>
<head>
<title>numbers.html</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Einfaches Zahlenspiel">

<style>

#game {
  display: block;
  width: 50%;
  color: white;
  padding: 20px;
  margin: 20px auto;
  overflow: hidden;
}

#display {
  background: #333;
  width: 1.5em;
  height: 1.2em;
  font-size: 80px;
  text-align: center;
  padding: 0.5em;
  margin: 0;
  border-radius: 10%;
  border: 1px solid #006;
  float: right;
}
```

```
#feedback {
  background: white;
  color: black;
  box-shadow: 0 0 5px white;
}

#restart-button {
  display: none;
}

.game-over #display {
  display: none;
}

.game-over #restart-button {
  display: block;
}

#game span {
  display: inline;
  width: 55px;
  float: left;
}

.exact {
  color: lime;
  font-weight: bold;
}

.exact:after {
  display: block;
  margin: -0.5em 0 0 1.5em;
  content: "r";
}

.inexact {
  color:red;
  font-weight: bold;
}

.inexact:after {
  display: block;
  margin: -0.5em 0 0 1.5em;
  content: "f";
}

button[class^="type"] {
  background-color: #FFF;
  width: 50px;
  height: 50px;
  color: #000;
  font-size: 120%;
  font-weight: bold;
  text-align: center;
  padding: 0.25em 0;
  margin: 0 5px 5px 0;
  border-radius: 40%;
  border: 1px solid #006;
  float: left;
}

:not(.game-over) [class^="type"] {
  cursor: pointer;
}

button.type1 {background: red; }
button.type2 {background: orange;}
button.type3 {background: yellow;}
button.type4 {background: lime;}
button.type5 {background: skyblue;}
button.type6 {background: blue; color: white;}
button.type7 {background: purple; color: white}
button.type8 {background: maroon; color: white;}
button.type9 {background: darkgreen; color: white;}

</style>
```

```
<script>
/*
  Spielaufbau:
  (1) Eine Zufallszahl wird als Additionsziel definiert.
  (2) In einem Zahlenfeld werden so lange Zahlen angeklickt und addiert
      bis die Zielzahl erreicht oder überschritten wird. Dann wird
      entweder der Erfolg (r) oder der Misserfolg (f) angezeigt.
      Nun wird eine neue Zufallszahl als Additionsziel definiert.
  (3) Das Spiel ist beendet, wenn alle Zahlen angeklickt worden sind,
      oder wenn die Zielzahl nicht mehr erreicht werden kann.
  Hinweis:
  Das Programm demonstriert das dynamische Erzeugen von HTML-Elementen
  und die unterschiedliche Verwendung von CSS-Selektoren:

  id-Selektoren (#id), Klassen-Selektoren (.class), Pseudo-Selektoren (:).
  Der Attribut-Selektor [attr^="wert"] wählt nur jene Elemente aus,
  deren Attribute "attr" mit dem Ausdruck "wert" beginnen.
  Der Pseudo-Selektor :not(.class) wählt nur jene Elemente aus, welche
  das nachfolgende Klassenattribut NICHT aufweisen.
  Die Pseudo-Selektoren .class:before oder .class:after fügen einen
  Inhalt (content) vor oder nach das Element mit dem Klassenselektor ein.
  Die Selektor-Attribute "display: inline", "display: block", "display: none"
  bestimmen, ob Elemente nebeneinander oder untereinander angezeigt werden.
  Andere Attribute bestimmen die Stilmerkmale der Elemente (Größe, Farbe, usw.)
  Alle diese Selektoren können miteinander kombiniert werden.
*/

  // Eventhandler, der automatisch sofort nach Ladung des HTML-Dokumentes
  // aufgerufen wird und zuerst die selbstauslösende, anonyme Setup-Funktion
  // ausführt.

  document.addEventListener("DOMContentLoaded", function() {

  // Variablenliste:
  var container = document.getElementById("game"),  // Das Spielfeld
      display = document.createElement("p"),        // Zielzahl-Anzeige
      restart = document.createElement("button"),   // Restart-Button
      buttons = [],      // Array aller Klickbuttons mit Klickzahlen (30)
      numbers = [],      // Array aller zufälligen Zielzahlen (10)
      size = 10,         // Anzahl diser Zielzahlen
      index = 0,         // Zeiger auf die aktuelle Zielzahl
      current = 0,       // Zwischensumme
      result = 0;        // Erreichte Treffer

  // Selbstauslösende, anonyme Setup-Funktion. Zuerst werden in einer
  // Schleife "size"-Mal zufällige Zielzahlen "r" erzeugt und in dem
  // Array "numbers" abgespeichert. Jede Zielzahl wird sodann in drei
  // kleinere Zahlen z1, z2, z3 zerlegt und aufgelistet. Mit Hilfe
  // dieser Liste werden die Klickbuttons mit den Klickzahlen erzeugt
  // und in dem Array "buttons" abgespeichert. Das ergibt dann eine
  // Anzahl von 10 * 3 = 30 Elementen.
  // Der erste Parameter in der inneren Funktion von "forEach" ist immer
  // das jeweilige Array-Element (hier eine Klickzahl). Der zweite,
  // optionale Parameter ist der Index des jeweiligen Array-Elementes.


  (function() {
     var i, r, z1, z2, z3;
     for (i = 0; i < size; i++)
     {
        r = rand(10,19);
        numbers.push(r);
        z1 = rand(3,Math.floor(r/2));
        z2 = rand(1,4);
        z3 = r - z1 - z2;
        [z1, z2, z3].forEach(function(z) {
             var b = document.createElement("button");
             b.innerHTML = z;
             b.className = "type" + rand(1,9);
             buttons.push(b);
        });
     }

     // Display und Restart-Button stylen

     display.id = "display";
     restart.id = "restart-button";
     restart.innerHTML = "Spiel starten";
```

```
    // Container leeren
    while (container.firstChild) {
       container.removeChild(container.firstChild);
    }

    // Click-Listener zum Container hinzufügen
    container.addEventListener("click", click);

    // Display-Element in den Container einhängen
    container.appendChild(display);

    // Restart-Button in Container einhängen
    container.appendChild(restart);

    // Container mit "Ende"-Markierung versehen
    container.classList.add("game-over");
}());

// Handler-Funktion zur Auswertung der Klickzahlen und des Restart-Buttons.
// Die 30 Buttons werden nur ausgewertet wenn das Spiel gestartet wurde.
// Der erste Parameter in der inneren Funktion von "forEach" ist immer
// das jeweilige Array-Element (hier ein Button). Der zweite, optionale
// Parameter ist der Index des jeweiligen Array-Elementes. In der Funktion
// wird zuerst die Klickzahl "z" ermittelt und dann die Zwischensumme
// "current" um diese Zahl verringert. Danach wird der entsprechende Button
// entfernt. Zuletzt wird überprüft, ob ein Treffer vorliegt oder nicht.
// Am Schluss wird mit einer Zeitverzögerung die Routine "next" aufgerufen.

function click(e) {
  if (e.target == restart) { return start(); }
  if (container.classList.contains("game-over") || current <= 0) { return; }

  buttons.forEach(function(b) {
    if (e.target == b)
    {
       var z = b.textContent;
       current -= parseInt(z);
       b.parentNode.removeChild(b);
       if (current <= 0)
       {
          display.className = ( current === 0 ? "exact" : "inexact" );
          if (display.className == "exact") { result++; }
          setTimeout(next, 2000);
       }
     }
  });
  e.preventDefault();
  e.stopPropagation();
}

// Funktion für neue/weitere Zielzahl oder Ende-Überprüfung.
// Zuerst wird die Restsumme "r" aller noch übrigen Buttons ermittelt.
// Der Attributselektor [attr^=wert] wählt ein Element aus dessen CSS-Attribut
// mit "wert" beginnt (hier alle Buttons mit dem Klassen-Attribut "type").
// Dann wird der Zeiger "index" auf die nächste Zielzahl gesetzt
// und diese ausgegeben.

function next() {
  var r = 0; // Restwert aller übrigen Buttons
  document.querySelectorAll('button[class^="type"]').forEach( function(b) {
     r += parseInt(b.textContent);
  });
  index--;
  if (index < 0 || r < numbers[index]) { return end(); }
  current = numbers[index];
  display.innerHTML = current;
  display.className = "";
}

// Funktion zum Beenden des Spiels.
// An das Container-Element wird ein Absatz-Element "p" angehängt,
// um das Endresultat des Spiels anzuzeigen

function end() {
  var p = container.appendChild(document.createElement("p"));
  p.innerHTML = "Du hast " + result + " von " + size + " Rechnungen richtig !";
  p.id = "feedback";
  container.className = "game-over";
  result = 0;
}
```

```
  // Hilfsfunktion für Integer-Zufallszahlen zwischen min und max.

  function rand(min,max) {
    var z = Math.floor(Math.random() * (max - min + 1)) + min; return z;
  }

  // Funktion zum Mischen von Array-Elementen.
  // Beginnend mit dem letzten Element werden die Array-Elemente absteigend
  // mit einem zufällig ausgewählten, darunter liegenden Element vertauscht.

  function shuffle(array) {
     var count = array.length, i, temp;
     while (count > 0)
     {
        count--;
        i = Math.floor(Math.random() * count);
        temp = array[count];
        array[count] = array[i];
        array[i] = temp;
     }
     return array;
  };

  // Funktion zum (erneuten) Starten des Spiels.
  // Zuerst Container leeren - bis auf Display und Restart.
  // Dann die 10 Zielzahlen "numbers" mischen und ihre Anzahl
  // in der Variablen "index" speichern.
  // Dann die 30 Klickschalter "buttons" mischen und sie in das
  // Spielfeld transportieren.
  // Diese werden in Tabellenform mit 5 ("max") Reihen
  // ("span" display: inline) und mit 6 Spalten angeordnet.
  // Zum Schluss wird das Element "game-over" entfernt und
  // die Funktion "next" aufgerufen.

  function start() {
    while (container.lastChild != display && container.lastChild != restart)
    {
      container.removeChild(container.lastChild);
    }
    shuffle(numbers);
    index = numbers.length;
    shuffle(buttons);
    var max = Math.floor(Math.sqrt(size*3)),
        i = 0,
        s;
    buttons.forEach(function(b,i) {
       if (i % max === 0) { s = container.appendChild(document.createElement("span")); }
       s.appendChild(b);
    });
    container.classList.remove("game-over");
    next();
  }

});

</script>
</head>

<body>
  <br>
  <h1>Numbers - ein einfaches Zahlenspiel</h1>
  <main>
    <p id="ansage">
       Drücke solange auf die Zahlen, bis ihre Summe die rechte Zufallszahl ergibt.<br>
       Es gibt insgesamt 10 Rechnungen. Das Spiel ist beendet, wenn alle Rechnungen<br>
       ausgeführt sind, oder die Zufallszahl nicht mehr erreicht werden kann.
    </p>
    <div id="game"> </div>
  </main>
</body>
</html>
```

Programmieren mit JavaScript, Teil 3
Grafiken mit dem CANVAS-Objekt

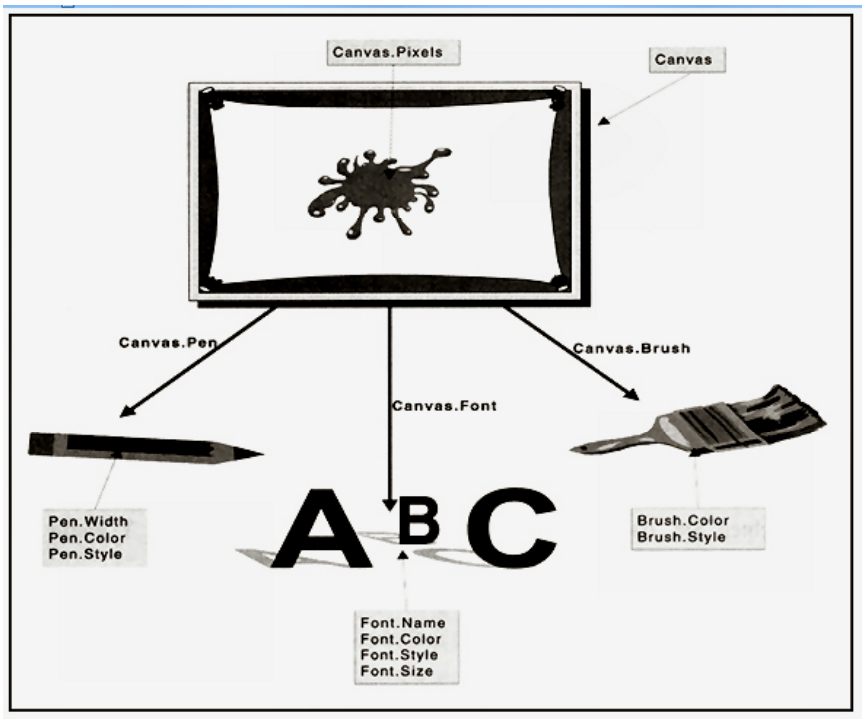
[3.1] Das canvas-Objekt

Das HTML-Element ***<canvas>*** ist eine Leinwand zur Darstellung von geometrischen Gebilden. Es ist ein DOM-Objekt mit vielen Eigenschaften und Methoden. Um es am Bildschirm genau zu lokalisieren, kann es von einem <*div*>-Tag eingeschlossen werden, dessen Position mit <*style*> angegeben wird. Dem Canvas werden zusätzlich eine ID und eine Breite (*width*) und Höhe (*height*) zugewiesen. Das Alles passiert in dem folgenden HTML-Programm sofort nach dem <*body*>-Tag. Alternative, direkte Positionierung des Canvas-Objektes mit Hilfe eines CSS-Style-Befehls:
<canvas style = "position: absolute; width: 500px; height: 500px; top: 60px; left: 600px;">

Um auf das Canvas zuzugreifen, muss dieses noch initialisiert werden, was dann in der Funktion ***initCanvas()*** geschieht. Diese kann als *onload*-Ereignis beim Starten des Programms aufgerufen werden. Um nun auf der Zeichenfläche tatsächlich zeichnen zu können, muss noch eine weitere Verbindung zur Canvas-Schnittstelle (Rendering Context) hergestellt werden. Diese kann beispielsweise durch eine Variable ***var ctx = document.getElementById("MyCanvas").getContex("2d")*** geschehen. Der Parameter ***("2d")*** legt eine zweidimensionale Zeichenfläche fest.

In dem Programm wird zusätzlich noch ein Ereignis am Canvas registriert, welches bei einem Absenken der Maus die physischen Pixel-Koordinaten eines Punktes auf der Zeichenfläche liefert. Dazu muss man wissen, dass der linke obere Randpunkt des Canvas die Koordinaten (0/0) hat und der rechte untere Randpunkt jene Koordinaten, welche durch ***width*** und ***height*** vorher definiert worden sind.

```
<!DOCTYPE html>
<html>
<head>
   <title> JS51 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <style type="text/css">
       body{background-color:#E0E8EF; color:black;
       font-family:Calibri; font-size:18px; font-weight:normal; text-size-adjust: none;
       text-align:left; margin:5%;}
       .cd1 {border:1px solid #666666; background-color:#FFFFFF; font-size:15px;}
   </style>

   <script>
    var canvas;   // globale Canvas-Variable
    var ctx;      // globale Kontext-Variable als Schnittstelle zum Canvas

    function initCanvas() {
       // Canvas initialisieren
       canvas = document.getElementById('MyCanvas');
       ctx = canvas.getContext('2d');
       ctx.lineWidth = 1;
       ctx.fillStyle = "white";
       ctx.fillRect(0,0,500,500);
       ctx.strokeStyle = "black";
       ctx.strokeRect(0,0,500,500);
    }

    function showPos(ev) {
       // Physische Canvas-Koordinaten ausgeben
       // ausgehend vom linken oberen Randpunkt
       x0 = ev.clientX - 600;
       y0 = ev.clientY - 60;
       info = x0 + '  /  ' + y0;
       document.MyForm.aus0.value = info;
    }
   </script>
</head>

<body onload = "initCanvas()">
<div id='dc' style="position:absolute; top:60px; left:600px ">
  <canvas id='MyCanvas' width='500' height='500' onmousedown = "showPos(event)">
     Your browser does not support HTML5 Canvas.
  </canvas>
</div>
<form name="MyForm">
<h3> Programm JS51: Mit JavaScript ein CANVAS definieren! </h3>
Ein Mausklick in die Grafik zeigt die physischen Punktkoordinaten:<br>
<input id="aus0" type="text"  value=" " size="16" class="cd1" readonly>
</form>
</body>
</html>
```

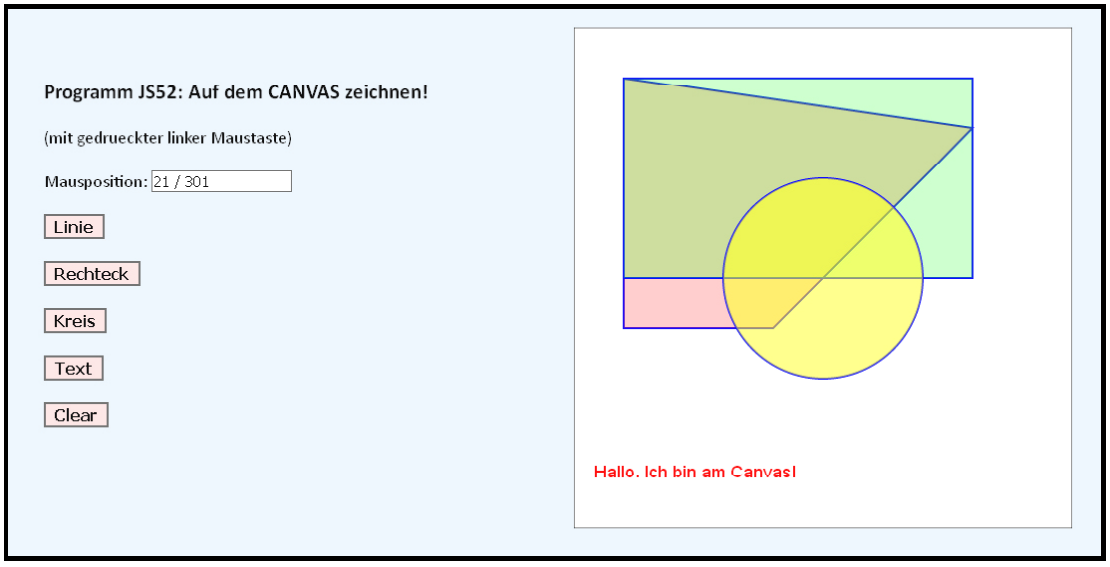


Symbolische Zeichenelemente: Pixels = Bildpunkte
Pen = Zeichenstift (stroke)
Brush = Pinsel (fill)
Font = Schriftsatz

In der Funktion ***initCanvas()*** stehen verschiedene Anweisungen für den Canvas:

```
ctx.font = '14px Calibri';          // Größe und Name des Schriftsatzes
ctx.lineWidth = 1;                  // Liniendicke
ctx.fillStyle = "white";            // Füllfarbe
ctx.fillRect(0,0,500,500);          // mit Farbe gefülltes Rechteck mit den diagonalen Eckpunkten
ctx.strokeStyle = "black";          // Linienfarbe
ctx.strokeRect(0,0,500,500);        // rechteckiger Linienrahmen
```

Im nachfolgenden Programm „js52.html“ wird auf dem Canvas mit gedrückter linker Maustaste gezeichnet. Zusätzlich können noch Linien, Rechtecke und Kreise gezeichnet werden.



```
<!DOCTYPE html>
<html>
<head>
   <title> JS52 </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">

   <style>
       body{background-color:#E0E8EF; color:black;
       font-family:Calibri; font-size:18px; font-weight:normal; text-size-adjust: none;
       text-align:left; margin:5%;}
       .cd1 {border:1px solid #666666; background-color:#FFFFFF; font-size:15px;}
       .cd2 {border:2px solid #666666; background-color:#EED8D8; font-size:16px;
             font-weight:bold;}
   </style>

   <script>

    var canvas;   // globale Canvas-Variable
    var ctx;      // globale Kontext-Variable als Schnittstelle zum Canvas
    var x,y;
    var go = false;

    function initCanvas() {
       // Canvas initialisieren
       canvas = document.getElementById('MyCanvas');
       ctx = canvas.getContext('2d');
       ctx.lineWidth = 1;
       ctx.fillStyle = "white";
       ctx.fillRect(0,0,500,500);
       ctx.strokeStyle = "black";
       ctx.strokeRect(0,0,500,500);
    }

    function drawLine() {
       ctx.lineWidth = 2;
       ctx.fillStyle = "rgba(255,0,0,0.25)";
       ctx.strokeStyle = "rgb(0,0,255)";
       ctx.beginPath();
       ctx.moveTo(50,50);
       ctx.lineTo(50,300);
       ctx.lineTo(200,300);
       ctx.lineTo(400,100);
       ctx.lineTo(50,50);
       ctx.stroke();
       alert("weiter");
       ctx.fill();
    }

    function drawRect() {
       ctx.lineWidth = 2;
       ctx.strokeStyle = "rgba(0,0,255,1.0)";
       ctx.strokeRect(50,50,350,200);
       alert("weiter");
       ctx.fillStyle = "rgba(0,255,0,0.25)";
       ctx.fillRect(50,50,350,200);
    }

    function drawCircle() {
       ctx.lineWidth = 2;
       ctx.strokeStyle = "rgba(0,0,255,1.0)";
       ctx.beginPath();
       ctx.arc(250,250,100,0,2*Math.PI);
       ctx.stroke();
       alert("weiter");
       ctx.fillStyle = "rgba(255,255,0,0.5)";
       ctx.fill();
    }

    function drawText() {
       ctx.font = "18px Arial";
       ctx.fillStyle = "red";
       ctx.fillText("Hallo. Ich bin am Canvas!",20,450);
    }

    function paint() {
      // Zeichnen ausführen
      if (go) {
         ctx.fillStyle = "red";
         ctx.fillRect(x,y,5,5);
      }
    }
```

```
    function move(ev) {
       x = ev.clientX - 600;
       y = ev.clientY - 60;
       var info = x + ' / ' + y;
       document.MyForm.aus0.value = info;
       paint();
    }
    function down() { go = true; }
    function up() { go = false; }

   </script>
</head>

<body onload = initCanvas()>
<div id='dc' style="position:absolute; top:60px; left:600px">
  <canvas id='MyCanvas' width='500' height='500'
     onmousedown=down() onmouseup=up() onmousemove=move(event)>
     Your browser does not support HTML5 Canvas.
  </canvas>
</div>
<form name="MyForm">
<br>
<h3> Programm JS52: Auf dem CANVAS zeichnen! </h3>
(mit gedrueckter linker Maustaste)<br>
<br>
Mausposition: <input id="aus0" type="text" value=" " size=16 class="cd1" readonly>
<br><br>
<input type="button" value="Linie" class="cd2" onClick="drawLine();"><br><br>
<input type="button" value="Rechteck" class="cd2" onClick="drawRect();"><br><br>
<input type="button" value="Kreis" class="cd2" onClick="drawCircle();"><br><br>
<input type="button" value="Text" class="cd2" onClick="drawText();"><br><br>
<input type="button" value="Clear" class="cd2" onClick="initCanvas();"><br><br>
</form>
</body>
</html>
```

In der nachfolgenden Tabelle sind die wichtigsten Eigenschaften und Methoden von ***canvas*** aufgelistet.

Property	Description
(•) Colors, Styles, and Shadows	
fillStyle	Sets or returns the color or pattern used to fill the drawing
strokeStyle	Sets or returns the color or pattern used for strokes
shadowColor	Sets or returns the color to use for shadows
shadowBlur	Sets or returns the blur level for shadows
shadowOffsetX	Sets or returns the horizontal distance of the shadow from the shape
shadowOffsetY	Sets or returns the vertical distance of the shadow from the shape
(•) Line Styles	
lineCap	Sets or returns the style of the end caps for a line
lineJoin	Sets or returns the type of corner created, when two lines meet
lineWidth	Sets or returns the current line width
(•) Rectangles	
rect()	Creates a rectangle
fillRect()	Draws a "filled" rectangle
strokeRect()	Draws a rectangle (no fill)
clearRect()	Clears the specified pixels within a given rectangle
(•) Text	
font	Sets or returns the current font properties for text content
textAlign	Sets or returns the current alignment for text content
fillText()	Draws "filled" text on the canvas
strokeText()	Draws text on the canvas (no fill)

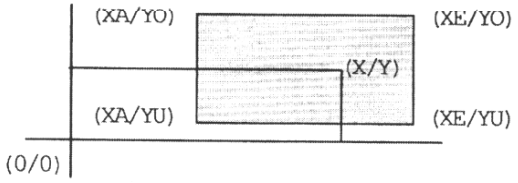
(●) Paths	
beginPath()	Begins a path, or resets the current path
closePath()	Creates a path from the current point back to the starting point
stroke()	Actually draws the path you have defined
fill()	Fills the current drawing (path)
moveTo()	Moves the path to the specified point in the canvas, without creating a line
lineTo()	Adds a new point and creates a line to that point from the last specified point in the canvas
arc()	Creates an arc/curve (used to create circles or parts of circles)
bezierCurveTo()	Creates a cubic Bézier curve
clip()	Clips a region of any shape and size from the original canvas
(●) Transformations	
scale()	Scales the current drawing bigger or smaller
rotate()	Rotates the current drawing
translate()	Remaps the (0,0) position on the canvas
transform()	Replaces the current transformation matrix for the drawing
setTransform()	Resets the current transform to the identity matrix. Then runs transform()
(●) Image Drawing	
drawImage()	Draws an image, canvas, or video onto the canvas
(●) Pixel Manipulation	
width	Returns the width of an ImageData object
height	Returns the height of an ImageData object
data	Returns an object that contains image data of a specified ImageData object
(●) Other	
save()	Saves the state of the current context
restore()	Returns previously saved path state and attributes
createEvent()	
getContext()	

Property	Description

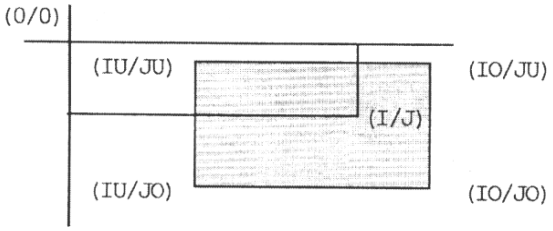
[3.2] Koordinaten-Transformation

Im Programm soll ein Weltbereich auf einen Bildbereich abgebildet werden. Der Weltbereich ist durch das Rechteck mit der Diagonale (XA/YO)–(XE,YU) begrenzt. Der Bildbereich ist durch das Rechteck mit der Diagonale (IU/JU)–(IO,JO) begrenzt.

(1) Der Weltbereich mit Diagonale (XA/YO)-(XE/YU)
Ein Weltpunkt (X/Y)



(2) Das Bildfenster mit Diagonale (IU/JU)-(IO/JO)
Ein Bildpunkt (I/J)



Es seien DX = XE-XA, DY = YO-YU, DI = IO-IU, DJ = JO-JU

Dann gelten folgende einfache proportionale Beziehungen:

(I) (I-IU) : +(X-XA) = DI : DX
(II) (J-JO) : -(Y-YU) = DJ : DY

Durch Umformung bekommt man folgende Transformationsformeln:

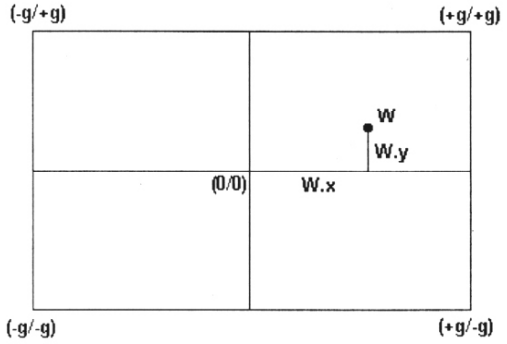
(I) I = +(X-XA) * DI/DX + IU
(II) J = -(Y-YU) * DJ/DY + JO

Der Weltbereich ist ein mathematisches Koordinatensystem. Der Bildbereich ist die Canvas-Komponente. Die Abmessungen in der Welt sind reelle Zahlen, während die Abmessungen im Bildbereich ganzzahlige Werte sind, welche die Anzahlen von Bildpunkten darstellen. Der Ursprung des Bildbereiches entspricht der linken oberen Ecke der Canvas-Komponente. Der Koordinaten-Ursprung des Weltbereichs liegt genau in der Mitte der Welt.

Der Sachverhalt vereinfacht sich wesentlich, wenn man als Weltbereich ein Quadrat mit der halben Seitenlänge **g** nimmt, also mit der Diagonale **(-g/+g)** – **(+g/-g)** begrenzt. Der Bildbereich ist durch die eingestellten Abmessungen der Canvas-Komponente (**0,0,Xmax,Ymax**) vorgegeben. Dabei ist **Xmax** die Bildbreite (width) und **Ymax** die Bildhöhe (height) des Canvas-Objektes.

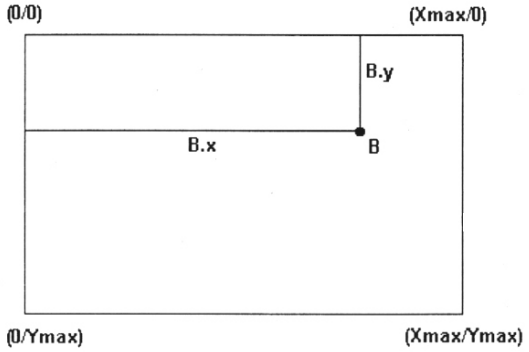
Ein Weltpunkt **W** hat die Weltkoordinaten **W.x**, **W.y** und der zugehörige Bildpunkt **B** hat die Bildkoordinaten **B.x**, **B.y**. Dann gelten geometrische Verhältnisse, welche die folgende Abbildung veranschaulichen soll. Alle Punkte sind in JavaScript als einfache Objekte **TPoint(x,y)** definiert. Siehe dazu auch Buchseite 121.

Weltbereich(-g,+g,+g,-g)



Der Koordinatenursprung (0/0) im Weltbereich liegt genau in der Mitte. Die positive X-Achse zeigt nach rechts und die positive Y-Achse hinauf. Die Diagonale geht von Punkt (-g/+g) nach (+g/-g).

Bildbereich(0,0,Xmax,Ymax)



Der Koordinatenursprung (0/0) im Bildbereich liegt in der linken, oberen Ecke. Die positive X-Achse zeigt nach rechts und die positive Y-Achse zeigt hinunter. Die Diagonale geht von Punkt (0/0) nach (Xmax/Ymax).

Die Weltbreite ***2g*** entspricht dabei der Bildbreite ***Xmax*** und die Welthöhe ***2g*** entspricht der Bildhöhe ***Ymax***. Wie aus der obigen Abbildung ersichtlich ist, bestehen zwischen den Weltkoordinaten *(W.x,W.y)* und den Bildkoordinaten *(B.x,B.y)* eines Punktes sehr einfache, direkt proportionale Beziehungen:

$$(W.x + g) : 2g = B.x : Xmax$$
$$(W.y + g) : 2g = (Ymax - B.y) : Ymax$$

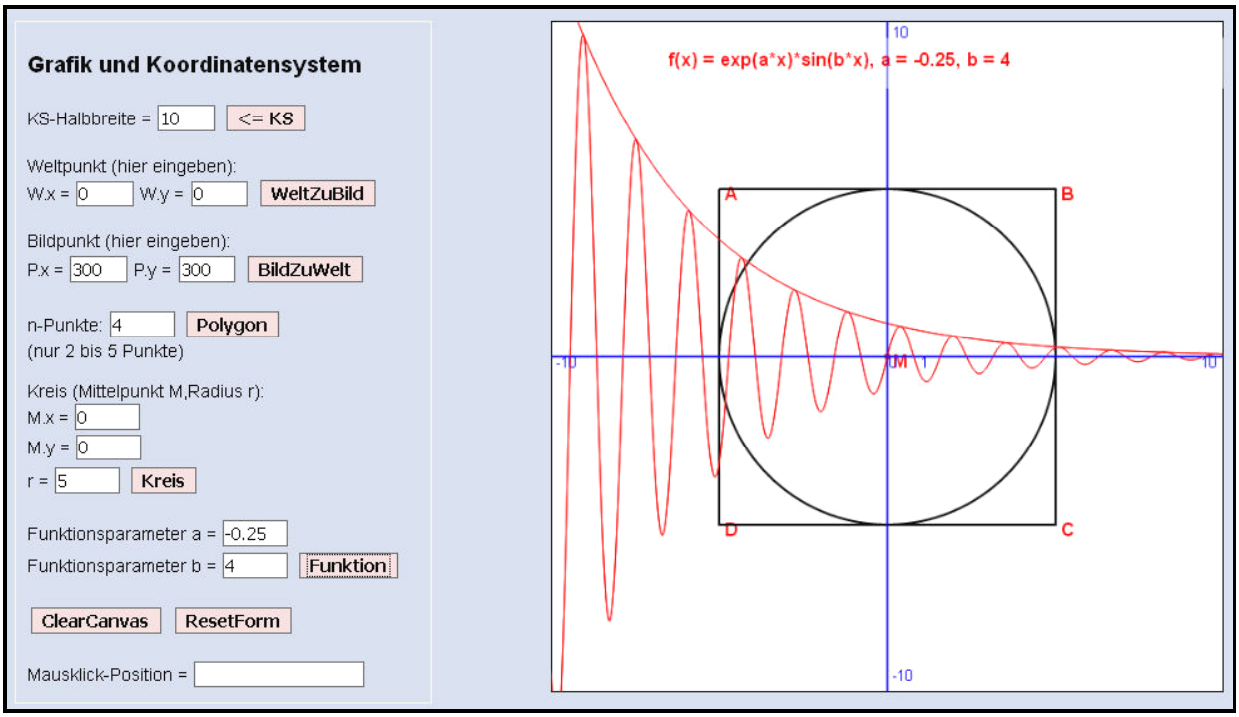
Durch Umformung bekommt man schließlich folgende Transformationsformeln:

```
B.x := Round( (W.x + g) * Xmax / (2*g) );
B.y := Round( Ymax - (W.y + g) * Ymax / (2*g) );
```

Im Folgenden soll der Bildbereich – so wie der Weltbereich – quadratisch sein, d.h. ***Ymax = Xmax.*** Die Funktion ***w2p(W, g, Xmax)*** berechnet zum gegebenen Weltpunkt ***W(W.x,W.y)*** den zugehörigen Bildpunkt ***B****(B.x,B.y).* Die Hilfsfunktion ***round2*** rundet auf zwei Dezimalstellen, und das Objekt ***TPoint*** repräsentiert die Punkte in der Ebene (siehe auch das nachfolgende Programm „js53.html“).

```
function w2p(W, g, Xmax) {
  B = new TPoint;
  B.x = round2((W.x + g) * Xmax / (2*g));
  B.y = round2(Xmax - W.y + g) * Xmax / (2*g));
  return B;
}
```

[3.2.1] Koordinaten-Transformation (js53.html)



Das Programm verwendet die oben beschriebenen Koordinaten-Transformationen. Der Bildbreich ist ein Canvas-Objekt, welches in einem div-Objekt eingebettet ist. Dabei erstreckt sich der quadratische Bildbereich von (0/0) bis (kb/kb) mit kb = 600. Der Weltbereich ist ein kartesiches Koordinatensystem von (-kw/kw) bis (kw/-kw), wobei kw die Halbbreite des Koordinatensystems ist. Sie ist standardmäßig 10, kann aber verändert werden, was eine Skalierung des Bildbereichs ermöglicht.

Die Transformationsformel **function w2p(W,kw,kb)** bildet einen Weltpunkt W auf den entsprechenden Bildpunkt P ab.

In dem obigen Screenshot wurde zuerst ein Kreis und und dann ein quadratisches Polygon gezeichnet. Zusätzlich wurden noch eine fallende Exponentialfunktion und eine gedämpfte Schwinung dargestellt. Maximal können 9 verschiedene Funktionen mit jeweils zwei Parametern dargestellt werden.

Das Programm enthält am Anfang den Funktioncode "**smart()**", der das Programmdesign an den kleinen Bildschirm von Smartphones anpasst (Responsives Webdesign), wobei

(1) die VIEWPORT-Skalierung passend abgeändert wird,
(2) die Grafik hinter den HTML-Body verlagert wird,
(3) einige Eigenschaften von einigen Objektklassen verändert werden.

Diese Anpassungs-Funktion wird am Ende zwischen </body> und </html> aufgerufen.

Hinweis: In den meisten Übungsprogrammen des Lehrbuchs ist diese Anpassungsfunktion aus Platzgründen nicht angeführt. Der interessierte Leser kann sie jedoch ohne Schwierigkeiten in den jeweilgen Programmcode einfügen.

```
<!DOCTYPE html>
<html>
<head>
  <title>js53.html</title>
  <meta charset="ISO-8859-1">
  <meta name="viewport" content="width=device-width, initial-scale=0.1">
  <meta name="author" content="Herbert Paukert">
  <meta name="description" content="Grafik und Koordinatensystem">
```

```
<style>
     body{background-color:#D0D8E8; color:black;
       font-family:'Arial'; font-size:16px; font-weight:normal; text-align:left; margin:20px;}
     #titel { font-size:20px; font-weight:bold;}
     .ctex  {border:1px solid #666666; background-color:#FFFFFF; margin: 2px; font-size:16px;}
     .cbtn  {border:2px solid #666666; background-color:#EED8D8; margin: 4px;
             font-size:16px; font-weight:bold;}
     fieldset { width: 350px;}
  </style>

  <script>
    var canvas;         // globale Canvas-Variable
    var ctx;            // globale Kontext-Variable als Schnittstelle zum Canvas
    var kw = 10;        // halbe Weltbreite (Koordinatensystem KS)
    var kb = 600;       // ganze Bildbreite

    var a = 1;          // Erster Funktions-Parameter
    var b = 1;          // Zweiter Funktions-Parameter
    var fNum = 0;       // Funktions-Nummer
    var lastNum = 0;    // Letzte Funktions-Nummer (wird immer gespeichert)
    var fSign = 1;      // Funktions-Vorzeichen
    var xMin = -kw;     // Startpunkt der Funktion
    var xMax = kw;      // Endpunkt der Funktion
    var xStep = 1/100;  // Schrittweite der Funktion
    var fCol = 'red';   // Farbe der Funktion
    var fText = ' ';    // Name der Funktion
    var fMax = 10;      // Maximalanzahl der Funktionen
    var fArr = new Array(fMax);  // Array der Funktionen

    var anz;            // eingegebene Anzahl der Array-Punkte
    var max = 5;        // maximale Anzahl der Array-Punkte
    var Points = new Array(max);           // Array der Punkte
    for (i = 1; i <= max; i++) { Points[i] = '0,0'; }

    var smallWidth = 800;
    var scrWidth = window.screen.width;
    var smallScr = false;
    if (scrWidth <= smallWidth) { smallScr = true; }

  function smart() {
  // Anpassung an Smartphones (RWD, Responsives Webdesign)
  // Dabei wird der Canvas ans Ende des Bodys platziert
    if (smallScr) {
       document.querySelector('meta[name="viewport"]').setAttribute('content', 'initial-scale=0.8');
       liste1 = document.getElementsByClassName("ctex");
       for(var i = 0; i < liste1.length; i++) {
          liste1[i].style.margin = "4px";
          liste1[i].style.fontSize = "20px";
       }
       liste2 = document.getElementsByClassName("cbtn");
       for(var i = 0; i < liste2.length; i++) {
          liste2[i].style.margin = "6px";
          liste2[i].style.fontSize = "20px";
       }
       document.getElementById("body").style.fontSize = 18 + "px";
       document.getElementById("body").style.margin = 10 + "px";
       document.getElementById("fs").style.width = 500 + "px";
       let end = document.body.offsetHeight;
       let el = document.getElementById("dc");
       if (el) {
         el.style.position = "absolute";
         el.style.top = end + 20 + "px";
         el.style.left = 10 + "px";
       }
    }
  }

  function round2(x) { return Math.round(100*x)/100 }

  function fun(x) {
    // Funktion festlegen
    fSign = 1;
    if (fNum == 0) { y = 0; }                                    // Keine-F.
    if (fNum == 1) { y = b*x + a; }                              // Linear-F.
    if (fNum == 2) { y = b*x*x + a; }                            // Parabel-F.
    if (fNum == 3) { fSign = -1; y = (b/a)*Math.sqrt(a*a - x*x); }  // Ellipsen-F.
    if (fNum == 4) { fSign = -1; y = (b/a)*Math.sqrt(x*x - a*a); }  // Hyperbel-F.
    if (fNum == 5) { y = b/(x-a); }                              // Rational-F.
    if (fNum == 6) { y = Math.exp(b*x) + a; }                    // Exponential-F.
    if (fNum == 7) { y = Math.log(b*x) + a; }                    // Logarithmus-F.
    if (fNum == 8) { y = a*Math.sin(b*x); }                      // Sinus-F.
    if (fNum == 9) { y = Math.exp(a*x)*Math.sin(b*x); }          // Schwinungs-F.
    return y;
  }
```

```
function TPoint(tx,ty) {
  // Konstruktor für das Punkt-Objekt
  this.x = tx;
  this.y = ty;
  this.len = function() {
     z = Math.sqrt(this.x*this.x + this.y*this.y);
     return z.toFixed(2);
   }
}

function w2p(W,kw,kb) {
  // WeltzuBild-Transformation von Weltpunkt W auf Bildpunkt P
  P = new TPoint;
  P.x = round2((W.x + kw) * kb / (2*kw));
  P.y = round2(kb - (W.y + kw) * kb / (2*kw));
  return P;
}

function p2w(P,kw,kb) {
  // BildzuWelt-Transformation von Bildpunkt P auf Weltpunkt W
  // Inverse Transformation zu "w2p(W,kw,kb)"
  W = new TPoint;
  W.x = round2(P.x*2*kw/kb - kw);
  W.y = round2((kb - P.y)*2*kw/kb - kw);
  return W;
}

function WtoP() {
  // Weltpunkt W eingeben und zu Bildpunkt P transformieren
  var W = new TPoint;
  W.x = 1*document.MyForm.aus1.value;
  W.y = 1*document.MyForm.aus2.value;
  var P = new TPoint;
  P = w2p(W,kw,kb);
  document.MyForm.aus3.value = P.x.toFixed(0);
  document.MyForm.aus4.value = P.y.toFixed(0);
  drawPoint("P",W);
  if (smallScr) { document.getElementById("dc").scrollIntoView(); }
}

function PtoW() {
  // Bildpunkt P eingeben und zu Weltpunkt W transformieren
  var P = new TPoint;
  P.x = 1*document.MyForm.aus3.value;
  P.y = 1*document.MyForm.aus4.value;
  var W = new TPoint;
  W = p2w(P,kw,kb);
  document.MyForm.aus1.value = W.x.toFixed(0);
  document.MyForm.aus2.value = W.y.toFixed(0);
  drawPoint("P",W);
  if (smallScr) { document.getElementById("dc").scrollIntoView(); }
}

function changeKS() {
  // Änderung der Halbbreite kw des Welt-Koordinatensystems
  var x = 1*document.MyForm.aus0.value;
  kw = x;
  xMin = -kw;
  xMax = kw;
  initCanvas();
}

function initCanvas() {
  // Canvas initialisieren
  canvas = document.getElementById('MyCanvas');
  ctx = canvas.getContext('2d');
  ctx.lineWidth = 2;
  ctx.fillStyle = "white";
  ctx.fillRect(0,0,kb,kb);
  ctx.strokeStyle = "black";
  ctx.strokeRect(0,0,kb,kb);
  drawCoord2(kw,kb);
  document.getElementById('aus9').value = '';
}

function drawCoord2(kw,kb) {
  // Koordinatensystem zeichnen
  var g,h;
  ctx.strokeStyle = "black";
  ctx.strokeRect(0,0,kb,kb);
  ctx.font = '10pt Arial';
```

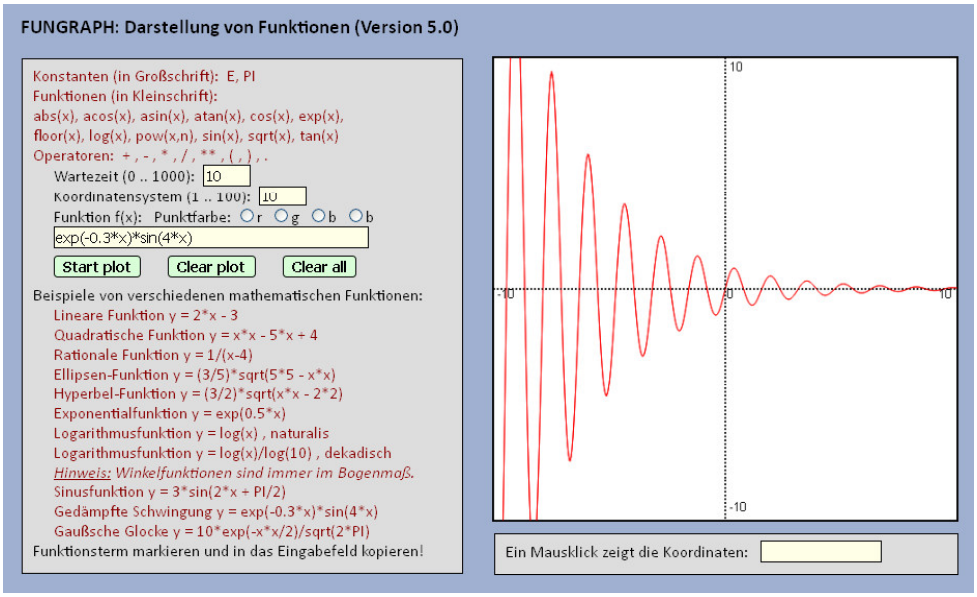
```
  ctx.beginPath();
    g = kb / 2;
    ctx.lineWidth = 1;
    ctx.strokeStyle = "blue";
    ctx.moveTo(0,g);
    ctx.lineTo(2*g,g);
    ctx.moveTo(g,0);
    ctx.lineTo(g,2*g);
    ctx.stroke();
    h = g / kw;
    ctx.fillStyle = "blue";
    ctx.fillText('0',g+2,g+10);
    ctx.fillText('1',g+h,g+10);
    ctx.fillText(-kw,5,g+10);
    ctx.fillText(kw,2*g-20,g+10);
    ctx.fillText(kw,g+5,15);
    ctx.fillText(-kw,g+5,2*g-10);
    ctx.strokeStyle = "black";
  }

  function drawPoint(tex,P) {
    // Punkt P zeichnen und mit tex beschriften
    var Q = new TPoint;
    Q = w2p(P,kw,kb);
    var r = 3;
    ctx.font = 'bold 14px Arial';
    ctx.fillStyle = "red";
    ctx.beginPath();
    ctx.arc(Q.x,Q.y,r,0,2*Math.PI);
    ctx.fill();
    drawPosText(P.x,P.y,tex)
}

  function drawPosText(x,y,tex) {
  // Text positioniert ausgeben
    ctx.font = 'bold 13pt Arial';
    var P = new TPoint;
    P.x = x;
    P.y = y;
    P = w2p(P,kw,kb);
    ctx.fillStyle = "red";
    ctx.fillText(tex,P.x+5,P.y+10);
  }

  function drawLine(P,Q) {
    //  Strecke durch die Punkte P und Q zeichnen
    var U = new TPoint;
    var V = new TPoint;
    U = P;
    V = Q;
    U = w2p(U,kw,kb);
    V = w2p(V,kw,kb);
    ctx.strokeStyle = "black"
    ctx.beginPath();
    ctx.moveTo(U.x,U.y);
    ctx.lineTo(V.x,V.y);
    ctx.stroke();
  }

  function drawCircle(M,r) {
    // Kreis mit Mittelpunkt M und Radius r zeichnen
    ctx.strokeStyle = "black";
    ctx.lineWidth = 2;
    var M1 = new TPoint;
    var P1 = new TPoint;
    var r1;

    M1 = M;
    M1 = w2p(M1,kw,kb);
    P1.x = r;
    P1.y = 0;
    P1 = w2p(P1,kw,kb);
    r1 = Math.abs(P1.x - kb/2);
    ctx.beginPath();
    ctx.arc(M1.x,M1.y,r1,0,2*Math.PI);
    ctx.closePath();
    ctx.stroke();
    drawPoint('M',M);
  }
```

```
function drawPoly() {
  // Polygon zeichnen
  var P = new TPoint;
  var Q = new TPoint;
  ctx.strokeStyle = "black";
  ctx.lineWidth = 2;
  ctx.beginPath();
  for (i = 1; i <= anz; i++) {
    z = Points[i];
    zerlegung = z.split(',');
    P.x = 1 * zerlegung[0];
    P.y = 1 * zerlegung[1];
    if ( isNaN(P.x) || isNaN(P.y) ) {
        alert('Abbruch: Falsche Koordinateneingabe.');
        return;
    }
    Q = w2p(P,kw,kb);
    if (i == 1) { ctx.moveTo(Q.x,Q.y); }
    ctx.lineTo(Q.x,Q.y);
    a = String.fromCharCode(64+i);
    if (i < anz) { drawPosText(P.x,P.y,a); }
  }
  ctx.stroke();
}

function WtoPoly() {
  // Polygon-Punkte eingeben und Polygon zeichnen
  anz = 1*document.MyForm.aus5.value;
  if ((anz < 2) || (anz > 5)) {
     alert('Abbruch: Falsche Punkteanzahl.');
     return;
  }
  var z = '0,0';
  Points[0] = z;
  for (i = 1; i <= anz; i++) {
     z = prompt(i +'-te Punktkoordinaten (x,y)',z);
     if (z.indexOf(',') < 0) {
        alert('Abbruch: Falsche Koordinateneingabe.');
        return;
     }
     Points[i] = z;
  }
  anz = anz + 1;
  Points[anz] = Points[1];
  drawPoly();
  if (smallScr) { document.getElementById("dc").scrollIntoView(); }
}

function WtoCirc() {
  // Kreis-Parameter M und r eingeben und Kreis zeichnen
  var M = new TPoint;
  M.x = 1*document.MyForm.aus6.value;
  M.y = 1*document.MyForm.aus7.value;
  r = 1*document.MyForm.aus8.value;
  drawCircle(M,r);
  if (smallScr) { document.getElementById("dc").scrollIntoView(); }
}

function drawFun(fSign,xMin,xMax) {
  // Funktion fun(x) mit dem Argument x zwischen xMin und xMax zeichnen.
  // Originale Funktion mit fSign = 1, gespiegelte Funktion mit fSign = -1.
  var x,x1,x2,y1,y2;
  var asymptot = false;
  ctx.lineWidth = 1;
  ctx.strokeStyle = fCol;
  var R = new TPoint;
  var S = new TPoint;
  xd = xStep;
  x = xMin;
  x1 = x + xd;
  do {
     asymptot = false;
     x = x + xd;
     x1 = x;
     y1 = fSign * fun(x1);
     x2 = x + 2*xd;
     y2 = fSign * fun(x2);
     // alert(x1 + ' / ' + y1 + '\n' + x2 + ' / ' + y2);
```

```
      if (y1 > 2*kw)    { y1 = kw;  asymptot = true; }
      if (y1 < (-2)*kw) { y1 = -kw; asymptot = true; }
      if (y2 > 2*kw)    { y2 = kw;  asymptot = true; }
      if (y2 < (-2)*kw) { y2 = -kw; asymptot = true; }
      if (Number.isNaN(y1)) { y1 = 0; }
      if (Number.isNaN(y2)) { y2 = 0; }
      R.x = x1;
      R.y = y1;
      R = w2p(R,kw,kb);
      S.x = x2;
      S.y = y2;
      S = w2p(S,kw,kb);
      if (asymptot || (y1 == 0 && y2 == 0)) { ctx.strokeStyle = "white"; }
      else { ctx.strokeStyle = fCol; }
      ctx.beginPath();
      ctx.moveTo(R.x,R.y);
      ctx.lineTo(S.x,S.y);
      ctx.stroke();
      ctx.closePath();
    } while (x <= xMax);
   drawCoord2(kw,kb);
}

function drawRect(x1,y1,x2,y2,farbe) {
  // Rechteck in der Grafik zeichnen.
  // Das wird zum Löschen des Schriftbereichs in der Grafik verwendet.
  var W1 = new TPoint;
  var W2 = new TPoint;
  var P1 = new TPoint;
  var P2 = new TPoint;
  W1.x = x1;
  W1.y = y1;
  W2.x = x2;
  W2.y = y2;
  P1 = w2p(W1,kw,kb);
  P2 = w2p(W2,kw,kb);
  ctx.fillStyle = farbe;
  ctx.fillRect(P1.x,P1.y,P2.x,P2.y);
}

function plotFun() {
// Auswahl einer Funktion mit den Parametern a und b
   a = 1*document.MyForm.ausA.value;
   b = 1*document.MyForm.ausB.value;
   if ((isNaN(a)) || (isNaN(b))) { alert('Abbruch - Parameterfehler!'); return; }
   fArr.length = 0;
   var r = lastNum;
   var s = '0: f(x) = 0, \n' +
           '1: f(x) = b*x + a, \n' +
           '2: f(x) = b*x*x + a, \n' +
           '3: f(x) = (b/a)*sqrt(a*a - x*x), \n' +
           '4: f(x) = (b/a)*sqrt(x*x - a*a), \n' +
           '5: f(x) = b/(x-a), \n' +
           '6: f(x) = exp(b*x) + a, \n' +
           '7: f(x) = log(b*x) + a, \n' +
           '8: f(x) = a*sin(b*x), \n' +
           '9: f(x) = exp(a*x)*sin(b*x), \n\n' +
           'Zuerst Parameter a und b, \n' +
           'dann Funktion eingeben. \n\n';
   fArr = s.split(',');
   var zahl = prompt(s,r);
   if (!isNaN(zahl)) {
     if ((zahl < 0) || (zahl > 9)) { zahl = 0; }
   }
   else { zahl = 0; }
   fNum = zahl;
   lastNum = fNum;
   fText = fArr[fNum].toString();
   t1 = fText.substr(4);
   t2 = 'a = ' + a + ', b = ' + b;
   t3 = t1.trim() + ', ' + t2;
   if (fNum > 0) {
      drawRect(-2*kw/3,kw-0.5,2*kw/3,kw-1.5,"white");
      drawPosText(-2*kw/3,kw-1,t3);
   }
   drawFun(1,xMin,xMax);
   if (fSign == -1) { drawFun(-1,xMin,xMax); }
   if (smallScr) { document.getElementById("dc").scrollIntoView(); }
}
```

```
  function showPos(ev) {
    // Angeklickte Koordinaten anzeigen
    var P = new TPoint;
    var cRect = document.getElementById("dc").getBoundingClientRect();
    var cTop  = cRect.top;
    var cLeft = cRect.left;
    P.x = (ev.clientX - cLeft);
    P.y = (ev.clientY - cTop);
    var W = new TPoint;
    W = p2w(P,kw,kb);
    var info = round2(W.x) + '  /  ' + round2(W.y);
    document.MyForm.aus9.value = info;
  }

  function resetForm() {
    // Formular initialisieren
    document.MyForm.reset();
    kw = 10;
    xMin = -kw;
    xMax = kw;
    fNum = 0;
    lastNum = 0;
    initCanvas();
    window.scrollTo(0,0);
  }

</script>
</head>

<body id="body" onload = "initCanvas();">
<div id='dc' style="position:absolute; top:20px; left:500px">
  <canvas id='MyCanvas' width='600' height='600' onmousedown='showPos(event);'></canvas>
</div>

<form name="MyForm">
<fieldset id="fs">
<p id="titel">Grafik und Koordinatensystem</p>
KS-Halbbreite = <input id="aus0" type="text" value="10" size="3" class="ctex">
<input type="button" value="<= KS" class="cbtn" onClick="changeKS()"><br>
<br>
Weltpunkt (hier eingeben):<br>
W.x = <input id="aus1" type="text" value="0" size="3" class="ctex">
W.y = <input id="aus2" type="text" value="0" size="3" class="ctex">
<input type="button" value="WeltZuBild" class="cbtn" onClick="WtoP()"><br>
<br>
Bildpunkt (hier eingeben):<br>
P.x = <input id="aus3" type="text" value="300" size="3" class="ctex">
P.y = <input id="aus4" type="text" value="300" size="3" class="ctex">
<input type="button" value="BildZuWelt" class="cbtn" onClick="PtoW()"><br>
<br>
n-Punkte: <input id="aus5" type="text" value="3" size="4" class="ctex">
<input type="button" value="Polygon" class="cbtn" onClick="WtoPoly()"><br>
(nur 2 bis 5 Punkte)<br><br>
Kreis (Mittelpunkt M,Radius r):<br>
M.x = <input id="aus6" type="text" value="0" size="4" class="ctex"><br>
M.y = <input id="aus7" type="text" value="0" size="4" class="ctex"><br>
r = <input id="aus8" type="text" value="5" size="4" class="ctex">
<input type="button" value="Kreis" class="cbtn" onClick="WtoCirc()"><br>
<br>
Funktionsparameter a = <input id="ausA" type="text" value="1" size="4" class="ctex"><br>
Funktionsparameter b = <input id="ausB" type="text" value="1" size="4" class="ctex">
<input type="button" value="Funktion" class="cbtn" onClick="plotFun()"><br>
<br>
<input type="button" value="ClearCanvas" class="cbtn" onClick="initCanvas()">
<input type="button" value="ResetForm" class="cbtn" onClick="resetForm()"><br>
<br>
Mausklick-Position = <input id="aus9" type="text" value=" " size="18" class="ctex">
</fieldset>
</form>

</body>
<script> smart(); </script>
</html>
```

[3.2.2] Darstellung von Funktionen OHNE Parameter (fungraph.html)



Hinweis: Das Programm verwendet in der Funktion „**Sleep**“ ein **„promise“-Objekt**. In der asynchronen Funktion „**drawFun**“ wird dann „**Sleep**“ aufgerufen, wodurch eine Zeitverzögerung erzeugt wird. Siehe dazu auch die Buchseite [130].

```
<!DOCTYPE html>
<html>
<head>
<title>fungraph.html</title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Funktionsplotter">

<style>
body {background-color:#A0B0D0; color:black; font-family:Calibri, sans-serif;
      font-size:18px; font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
.cd1 {border: 1px solid #000; background-color: #FFFFE8; font-size: 16px;}
.cd2 {border: 2px solid #000; background-color: #D8FFD8; font-size: 16px; font-weight: bold;
      border-radius: 4px; margin: 8px; }
#txt {color: darkred; }
#fun {color: darkred; }
#MyCanvas {border: 2px solid #000; }
#fs1 {width: 450px; background-color: #DDDDDD; border: 2px solid #000;}
#fs2 {position: absolute; left: 570px; top: 590px; width: 475px;
      background-color: #DDDDDD; border: 2px solid #000;}
</style>

<script>
  var canvas;          // globale Canvas-Variable
  var ctx;             // globale Kontext-Variable als Schnittstelle zum Canvas
  var farbe = 'red';   // Kurvenfarbe
  var kw = 10;         // halbe Weltbreite
  var kb = 500;        // ganze Bildbreite
  var mf = [];         // Array aller zulässigen Mathematikfunktionen (ohne Präfix "MATH.")
  mf[0]='abs'; mf[1]='acos'; mf[2]='asin'; mf[3]='atan'; mf[4]='cos'; mf[5]='exp';
  mf[6]='floor'; mf[7]='log'; mf[8]='pow'; mf[9]='sin'; mf[10]='sqrt'; mf[11]='tan';
  mf[12]='E'; mf[13] = 'PI';

  var s;   // Formel-String "s", dessen Berechnung mit "eval(s)" erfolgt, wobei im Unterprogramm
           // "Plot()" mittels regulärer Stringroutinen vor alle Mathematikfunktionen das Präfix
           // "MATH." eingefügt wird. Die nachfolgende Funktion "fun(s,x)" hat als ersten Parameter
           // den richtig umgeformten Formel-String "s" und als zweiten Parameter das Argument x.
  function fun(s,x) { return eval(s); }
  function round2(x) { return Math.round(100*x)/100 }
  var stop = false;
  var zeit = 10;
  function Sleep(msec) {
  // Wartefunktion in Millisekunden
     return new Promise(resolve => setTimeout(resolve,msec));
  }
```

```
function TPoint(tx,ty) {
  // Konstruktor für das Punkt-Objekt
  this.x = tx;
  this.y = ty;
  this.len = function() {
     z = Math.sqrt(this.x*this.x + this.y*this.y);
     return z.toFixed(2);
   }
}

function w2p(W,kw,kb) {
  // WeltzuBild-Transformation von Weltpunkt W auf Bildpunkt P
  P = new TPoint;
  P.x = round2((W.x + kw) * kb / (2*kw));
  P.y = round2(kb - (W.y + kw) * kb / (2*kw));
  return P;
}

function p2w(P,kw,kb) {
  // BildzuWelt-Transformation von Bildpunkt P auf Weltpunkt W
  // Inverse Transformation zu "w2p(W,kw,kb)"
  W = new TPoint;
  W.x = round2(P.x*2*kw/kb - kw);
  W.y = round2((kb - P.y)*2*kw/kb - kw);
  return W;
}

function initCanvas() {
  // Canvas initialisieren
  canvas = document.getElementById('MyCanvas');
  ctx = canvas.getContext('2d');
  ctx.lineWidth = 1;
  ctx.fillStyle = "white";
  ctx.fillRect(0,0,500,500);
  ctx.strokeStyle = "black";
  ctx.strokeRect(0,0,500,500);
  drawCoord2(kw,kb);
}

function drawCoord2(kw,kb) {
  // Koordinatensystem zeichnen
  var g,h;
  ctx.font = '10pt sans-serif';
  ctx.strokeStyle = "black";
  ctx.strokeRect(0,0,kb,kb);
  ctx.beginPath();
  ctx.lineWidth = 1;
  g = kb / 2;
  ctx.setLineDash([2,2]);
  ctx.strokeStyle = "black";
  ctx.moveTo(0,g);
  ctx.lineTo(2*g,g);
  ctx.moveTo(g,0);
  ctx.lineTo(g,2*g);
  ctx.stroke();
  ctx.setLineDash([]);
  h = g / kw;
  ctx.fillStyle = "black";
  ctx.fillText('0',g+2,g+10);
  ctx.fillText(-kw,5,g+10);
  ctx.fillText(kw,2*g-20,g+10);
  ctx.fillText(kw,g+5,15);
  ctx.fillText(-kw,g+5,2*g-10);
  ctx.strokeStyle = "black";
}

async function drawFun(kw,kb,s) {
  // kw = Halbbreite des Koordinatensystems, kb = Canvas-Bildbreite
  // Funktion fun(s,x) zeichnen, s = Funktionsterm, x = Argument
  var x,xd,x1,x2,y1,y2,xMin,xMax;
  var asymptot = false;
  ctx.lineWidth = 1;
  ctx.strokeStyle = farbe;
  var R = new TPoint;
  var S = new TPoint;
  xMin = -kw;
  xMax = kw;
  x = xMin;
  xd = 2*kw/kb;
  x = x - xd;
```

```
    do {
       asymptot = false;
       x = x + xd;
       x1 = x;
       y1 = fun(s,x1);
       x2 = x + 2*xd;
       y2 = fun(s,x2);
       if (y1 > 2*kw)    { y1 = kw;  asymptot = true; }
       if (y1 < (-2)*kw) { y1 = -kw; asymptot = true; }
       if (y2 > 2*kw)    { y2 = kw;  asymptot = true; }
       if (y2 < (-2)*kw) { y2 = -kw; asymptot = true; }
       if (Number.isNaN(y1)) { y1 = 0; }
       if (Number.isNaN(y2)) { y2 = 0; }
       R.x = x1;
       R.y = y1;
       R = w2p(R,kw,kb);
       S.x = x2;
       S.y = y2;
       S = w2p(S,kw,kb);
       if (asymptot || (y1 == 0 && y2 == 0)) { ctx.strokeStyle = "white"; }
       else { ctx.strokeStyle = farbe; }
       ctx.beginPath();
       ctx.moveTo(R.x,R.y);
       ctx.lineTo(S.x,S.y);
       ctx.stroke();
       ctx.closePath();
       if (zeit > 0) { await Sleep(zeit); }
       if (stop) { break; }
    } while (x <= xMax);
    drawCoord2(kw,kb);
  }

  function Plot() {
  // Zuerst Formel-String "s" richtig auswerten und dann die Funktion zeichnen
    kw = Math.abs(document.MyForm.ein1.value.trim());
    if (isNaN(kw) || (kw < 1) || (kw > 1000)) { kw = 10; document.MyForm.ein1.value = 10; }
    zeit = 1 * document.MyForm.ein0.value.trim();
    if (isNaN(zeit) || (zeit < 0) || (zeit > 100)) { zeit = 10; document.MyForm.ein0.value = 10; }
    s = document.MyForm.ein2.value.trim();
    for (var j = 0; j <= 13; j++) {            // Automatisches Einfügen von "Math."
        r = mf[j];                             // mithilfe regulärer Stringroutinen
        if (s.indexOf(r) > -1) {
           t = 'Math.' + r;
           reg = new RegExp(r,'gi');
           s = s.replace(reg,t);
        }
    }
    u = 'aMath.';                              // Berichtigung der Arcusfunktionen
    v = 'a';                                   // asin, acos und atan
    reg1 = new RegExp(u,'gi');
    s = s.replace(reg1,v);
    var btn = document.getElementsByName("col");
    for (var i = 0; i < btn.length; i++) {
        if (btn[i].checked) { farbe = btn[i].value; }
    }
    stop = false;
    drawCoord2(kw,kb);                         // Koordinaten-Achsen zeichnen
    drawFun(kw,kb,s);                          // Grafische Darstellung der Funktion
    document.MyForm.ein3.value = '';
  }

  function clearPlot() {
  // Grafik löschen
    kw = Math.abs(document.MyForm.ein1.value);
    initCanvas();
    document.MyForm.ein3.value = '';
    stop = true;
  }

  function clearAll() {
  // Alles löschen
    zeit = 10;
    document.MyForm.ein0.value = zeit;
    kw = 10;
    document.MyForm.ein1.value = kw;
    initCanvas();
    document.MyForm.ein2.value = '';
    document.MyForm.ein3.value = '';
    stop = true;
  }
</script>
</head>
```

```
<body onload="initCanvas();">

<div id='dc' style="position:absolute; top:75px; left:570px;">
  <canvas id='MyCanvas' width='500' height='500' onmousedown = "showPos(event)">
     Your browser does not support HTML5 Canvas.
  </canvas>
</div>

<form name="MyForm">
<h3>FUNGRAPH: Darstellung von Funktionen (Version 5.0)</h3>
<fieldset id="fs1">

<span id="txt">
Konstanten (in Großschrift):  E, PI <br>
Funktionen (in Kleinschrift):<br>
abs(x), acos(x), asin(x), atan(x), cos(x), exp(x),<br>
floor(x), log(x), pow(x,n), sin(x), sqrt(x), tan(x)<br>
Operatoren:  + , - , * , / , ** , ( , ) , .<br>
</span>

     Wartezeit (0 .. 1000):  <input type="text" name="ein0" value="10"
                                                    size="3" class="cd1"><br>
     Koordinatensystem (1 .. 100):  <input type="text" name="ein1"
                                                          value="10" size="3" class="cd1"><br>
     Funktion f(x):
  Punktfarbe:
<input type="radio" name="col" value="red">r 
<input type="radio" name="col" value="green">g 
<input type="radio" name="col" value="blue">b 
<input type="radio" name="col" value="black">b 
  
<br>
     <input type="text" name="ein2" value="exp(-0.3*x)*sin(4*x)" size="46"
                         class="cd1">
<br>
   <input type="button" name="rech1" value="Start plot" class="cd2"
              onclick="Plot()">  
<input type="button" name="rech2" value="Clear plot" class="cd2" onclick="clearPlot()">  
<input type="button" name="rech3" value="Clear all"  class="cd2" onclick="clearAll()">
<br>
Beispiele von verschiedenen mathematischen Funktionen:<br>

<span id="fun">
     Lineare Funktion y = 2*x - 3 <br>
     Quadratische Funktion y = x*x - 5*x + 4 <br>
     Rationale Funktion y = 1/(x-4) <br>
     Ellipsen-Funktion  y = (3/5)*sqrt(5*5 - x*x) <br>
     Hyperbel-Funktion  y = (3/2)*sqrt(x*x - 2*2) <br>
     Exponentialfunktion y = exp(0.5*x) <br>
     Logarithmusfunktion y = log(x) , naturalis <br>
     Logarithmusfunktion y = log(x)/log(10) , dekadisch <br>
     <i><u>Hinweis:</u> Winkelfunktionen sind immer im Bogenmaß.</i></b> <br>
     Sinusfunktion y = 3*sin(2*x + PI/2) <br>
     Gedämpfte Schwingung y = exp(-0.3*x)*sin(4*x) <br>
     Gaußsche Glocke y = 10*exp(-x*x/2)/sqrt(2*PI) <br>
</span>
Funktionsterm markieren und in das Eingabefeld kopieren!
</fieldset>

<fieldset id="fs2">
Ein Mausklick zeigt die Koordinaten:  
<input type="text" name="ein3" value=" " size="15" class="cd1" readonly="true">
</fieldset>
</form>

<script>
   function showPos(ev) {
     kw = 1 * document.MyForm.ein1.value;
     var x1 = window.pageXOffset + ev.clientX - 570;
     var y1 = window.pageYOffset + ev.clientY - 75;
     var x0 = x1*2*kw/kb - kw;
     var y0 = (kb - y1)*2*kw/kb - kw;
     var info = round2(x0) + '  /  ' + round2(y0);
     document.MyForm.ein3.value = info;
   }
</script>

</body>
</html>
```

[3.3] Vier Mathematikprogramme

Im Folgenden werden vier Aufgaben aus der Mathematik vorgestellt, die das Canvas-Objekt zu grafischen Darstellungen verwenden.

(1) Koch-Kurven (koch.html)

(2) Geometrische Abbildungen (abbild.html)

(3) Trigonometrie des Dreiecks (js59.html)

(4) Dreieck und Schwerpunkt (js60.html)

Alle vier Programme verwenden die externe JavaScript-Datei „**graph.js**“. Diese besteht aus einer Sammlung von globalen Variablen und Funktionen für das Canvas-Objekt. Der gesamte Quellcode dieser Datei ist am Ende des Kapitels aufgelistet.

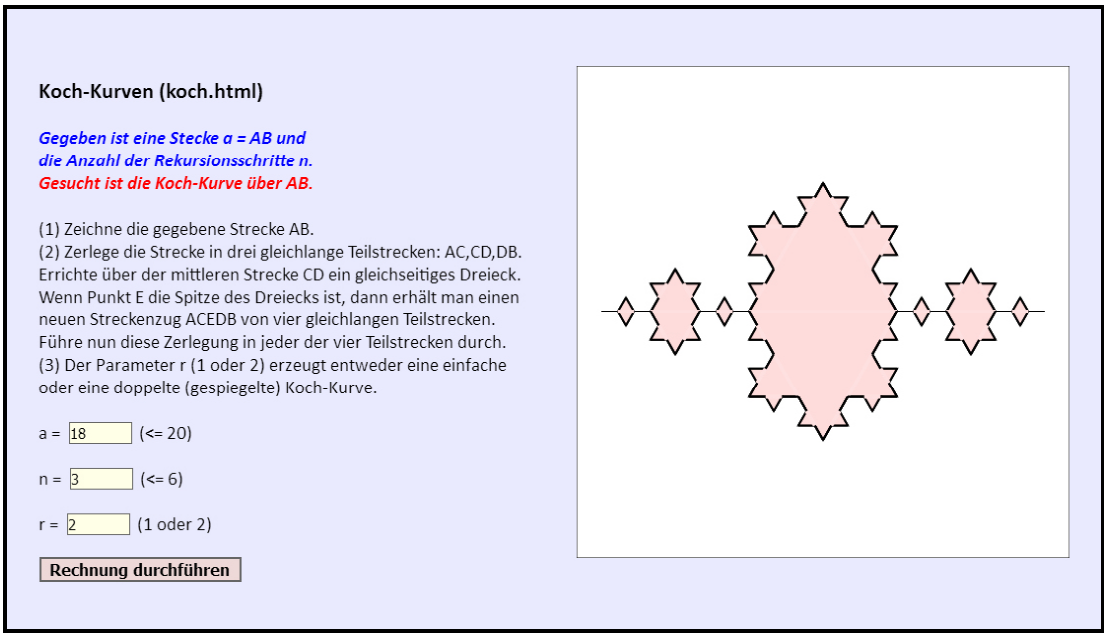
Das erste Programm **(koch.html)** verwendet zusätzlich ein so genanntes **„promise“-Objekt:**

Normalerweise ist der Programmablauf ***synchron***, d.h. jeder Befehl wird erst dann ausgeführt, wenn die Ausführung des vorangehenden Befehls beendet ist. Soll aber eine Funktion unabhägig vom synchronen Ablauf (***asynchron***) ausgeführt werden, so müssen vor ihre Funktionsanweisung das Schlüsselwort „**async**“ gestellt und zusätzlich noch ein „**promise**“ -Objekt definiert werden. Ein so genanntes promise-Objekt ist ein Objekt, das den möglichen Erfolg oder Misserfolg einer asynchronen Operation repräsentiert – im Fall des Erfolges mit der callback-Methode „**resolve**“, andernfalls bei Misserfolg mit der callback-Methode „**reject**“.

Auf den return-Wert solcher asynchronen Funktionen kann mit **.then** zugegriffen werden.
Der Aufruf des promise-Objektes in einer asynchronen Funktion kann mit „**await**“ erfolgen.
Von den folgenden Beispielen realisiert das letzte Beispiel eine Funktion „**sleep(msec)**“, welche den Ablauf einer asynchronen Funktion für msec Millisekunden unterbricht.

```
<script>
// erstes Beispiel ***********************************
var keepProm = true;
var Prom = new Promise(function(resolve, reject) {
    if (keepProm) { resolve = alert('The given promise IS KEPT !'); }
    else { reject = alert('The given promise IS NOT KEPT !'); }
});
Prom.resolve;  // liefert hier einen Wert
Prom.reject;   // liefert hier KEINEN Wert

// zweites Beispiel ***********************************
async function funA(x) {
  return Promise.resolve(Math.pow(x,2));
}
funA(4).then(alert);

// drittes Beispiel ***********************************
function sqr(x) {
   var Prom = new Promise(resolve => resolve(Math.pow(x,2)));
   return Prom;
}
async function funB(x) {
  alert(await sqr(x));
}
funB(5);

// viertes Beispiel ***********************************
function sleep(msec) {
   return new Promise(resolve => setTimeout(resolve,msec));
}
async function funC(zeit) {
  for (var x = 0; x <= 10; x++) {
    await sleep(zeit);
    var y = Math.pow(x,2);
    document.write('Das Quadrat von ',x,' ist ',y,'<br>');
  }
}
funC(1000);
</script>
```

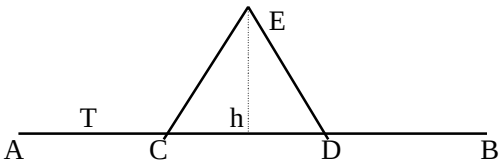
[3.3.1] Koch-Kurven (koch.html)



Eine Programmfunktion, die mit Hilfe eines entsprechenden Schaltknopfes aufgerufen werden kann, erzeugt durch **rekursive Programmiertechnik** ein schneeflockenartiges Gebilde.

Im vorliegenden **Demonstrationsprogramm** soll eine gegebene Strecke AB in drei gleich lange Teile AC, CD, DB zerlegt werden. Über dem mittleren Teil CD ist ein gleichseitiges Dreieck zu errichten, dessen Spitze der Punkt E ist.

Dieser Prozess wird dann rekursiv in jeder der so erzeugten vier gleich langen Strecken weitergeführt, bis eine vorgegebene Rekursionsstufe erreicht ist. Das Programm erzeugt also immer feiner werdende Streckenzüge, die einem schneeflockenartigen Gebilde ähneln, das auch als KOCH-Kurve bezeichnet wird. Die Berechnung der Zwischenpunkte C, E, D aus den Endpunkten A, B einer Basisstrecke erfolgt mit den Hilfsmitteln der elementaren Vektorgeometrie:



Die Punkte A,B,C,E,D und der Vektor T sind allesamt vom vordefinierten Datentyp *TPoint*.

Der Hilfsvektor **T** sei ein Drittel des Basisvektors **AB**. Dann gilt: T.x = (B.x-A.x)/3 und T.y = (B.y-A.y)/3. Für den Normalvektor **N** gilt: N.x = T.y und N.y = -T.x.
Die Höhe **h** im gleichseitigen Dreieck kann nach der Formel h = s/2*Math.sqrt(3) berechnet werden, wobei **s** die Dreiecksseite (also genau die Länge von T bzw. N) ist.
Damit gelten folgende Vektorgleichungen:

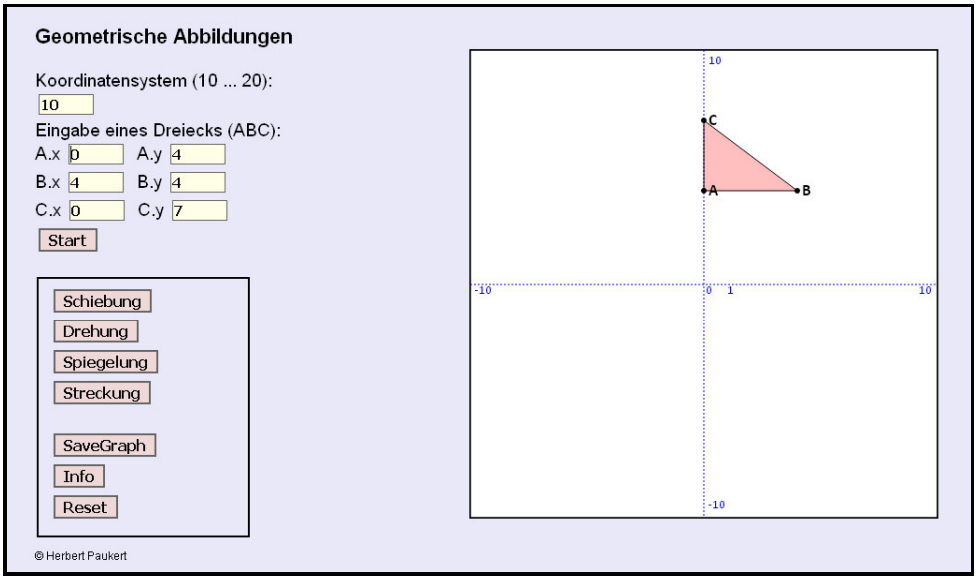
```
C = A + T;
D = C + T;
E = (A+B)/2 + N*Math.sqrt(3)/2;
```

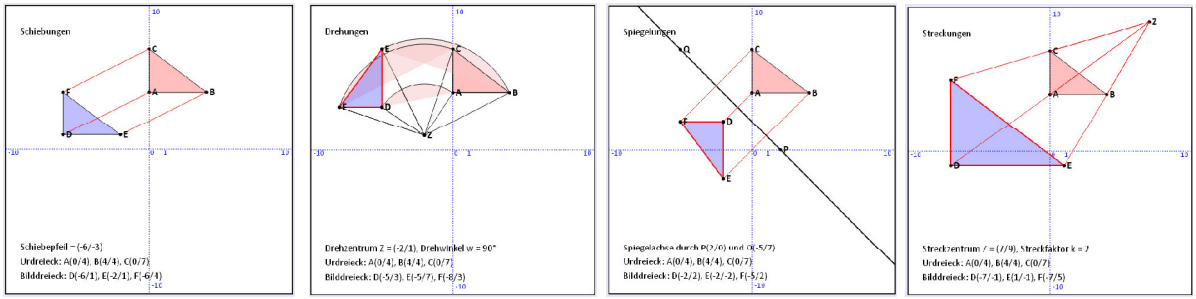
Daraus können unschwer die entsprechenden Koordinatenformeln gewonnen werden.

```
<!DOCTYPE html>
<html>
<head>
<title> Koch-Kurve </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Koch-Kurve">

<style>
body {background-color:#EAEAFF; color:black; font-family:Calibri,Arial;
      font-size:19px; font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
.cd1 {border: 1px solid #666666; background-color: #FFFFE8; font-size: 15px;}
.cd2 {border: 2px solid #666666; background-color: #EED8D8; font-size: 16px; font-weight: bold;}
</style>

<script src = "graph.js"></script>

<script>
function Sleep(msec) {
// Wartefunktion in Millisekunden
  return new Promise(resolve => setTimeout(resolve,msec));
}
async function koch(R,S,d,e) {
// rekursives (und asynchrones) Unterprogramm
  var T = new TPoint;
  var U = new TPoint;
  var V = new TPoint;
  var W = new TPoint;
  var N = new TPoint;

  T.x = (S.x - R.x) / 3;
  T.y = (S.y - R.y) / 3;
  U.x = R.x + T.x;
  U.y = R.y + T.y;
  V.x = U.x + T.x;
  V.y = U.y + T.y;
  N.x = U.x + T.x / 2;
  N.y = U.y + T.y / 2;
  W.x = N.x - e * T.y * Math.sqrt(3) / 2;
  W.y = N.y + e * T.x * Math.sqrt(3) / 2;

  drawColor("#000000");
  drawWidth(2);
  drawLine(R,U);
  drawLine(U,W);
  drawLine(W,V);
  drawLine(V,S);

  fillColor("rgba(255,208,208,0.75)");
  drawLine3(U,V,W);
  drawColor("#FEE0E0");
  drawWidth(4);
  drawLine(U,V);

  d = d - 1;
  if (d <= 0) { return; }

  await Sleep(200);

  koch(R,U,d,e);
  koch(U,W,d,e);
  koch(W,V,d,e);
  koch(V,S,d,e);
}

function Start() {
// Hauptprogramm
  var a = 1 * document.MyForm.ein1.value;
  var n = 1 * document.MyForm.ein2.value;
  var r = 1 * document.MyForm.ein3.value;

  if ( isNaN(a) || isNaN(n) || isNaN(r) ) {
     alert('Ungültige Eingaben !');
     return;
  }
  if (a <= 0) { a = 20; document.MyForm.ein1.value = a; }
  if (a > 20) { a = 20; document.MyForm.ein1.value = a; }
  if (n <= 0) { n = 1;  document.MyForm.ein2.value = n; }
  if (n > 6)  { n = 6;  document.MyForm.ein2.value = n; }
  if (r < 1)  { r = 1;  document.MyForm.ein3.value = r; }
  if (r > 2)  { r = 2;  document.MyForm.ein3.value = r; }
```

```
  var s = a / 3;
  var h = Math.sqrt(3) * s / 2;

  var A = new TPoint;
  var B = new TPoint;
  var C = new TPoint;
  var D = new TPoint;
  var E = new TPoint;
  var M = new TPoint;
  var N = new TPoint;

  A.x = -a/2;
  A.y = 0;
  B.x = a/2;
  B.y = 0;
  C.x = -s/2;
  C.y = 0;
  D.x = s/2;
  D.y = 0;
  M.x = 0;
  M.y = 0;
  E.x = M.x;
  E.y = M.y + h;

  initCanvas(kb,2,0);
  drawColor("#000000");
  drawWidth(2);
  fillColor("rgba(250,200,200,0.75)");
  drawLine(A,C);
  drawLine(C,E);
  drawLine(E,D);
  drawLine(D,B);

  if (r == 1) { koch(A,B,n,1); }
  if (r == 2) {
     koch(A,B,n,1);
     koch(A,B,n,-1);
  }
}

</script>
</head>

<body onload="Start()">
<div id='dc' style="position:absolute; top:60px; left:600px ">
     <canvas id='MyCanvas' width='500' height='500'>
     Your browser does not support HTML5 Canvas.
  </canvas>
</div>
<form name="MyForm">
<br>
<h3>Koch-Kurven (koch.html)</h3>
<b><i><font color = "blue">
Gegeben ist eine Stecke a = AB und<br>
die Anzahl der Rekursionsschritte n.<br>
</font></i></b>
<b><i><font color = "red">
Gesucht ist die Koch-Kurve über AB.<br>
</font></i></b>
<br>
(1) Zeichne die gegebene Strecke AB.<br>
(2) Zerlege die Strecke in drei gleichlange Teilstrecken: AC,CD,DB.<br>
Errichte über der mittleren Strecke CD ein gleichseitiges Dreieck.<br>
Wenn Punkt E die Spitze des Dreiecks ist, dann erhält man einen<br>
neuen Streckenzug ACEDB von vier gleichlangen Teilstrecken.<br>
Führe nun diese Zerlegung in jeder der vier Teilstrecken durch.<br>
(3) Der Parameter r (1 oder 2) erzeugt entweder eine einfache<br>
oder eine doppelte (gespiegelte) Koch-Kurve.<br>
<br>
a = <input type="text" name="ein1" value="20" size="5" class="cd1">  (&lt= 20)<br><br>
n = <input type="text" name="ein2" value="1" size="5" class="cd1">  (&lt= 6)<br><br>
r = <input type="text" name="ein3" value="1" size="5" class="cd1">  (1 oder 2)<br><br>
<input type="button" name="rech" value="Rechnung durchführen" class="cd2" onclick="Start()">
</form>
</body>
</html>
```

[3.3.2] Geometrische Abbildungen (abbild.html)





Das Programm enthält die Funktion "**smart()**", die das Programmdesign an Smartphones anpasst, wobei (1) die VIEWPORT-Skala geändert wird, (2) die Grafik hinter den HTML-Body verlagert wird und (3) die Schrift-Größe in einigen Objektklassen verändert wird (siehe auch Seite 121).

Eine Schiebung ist durch den Schiebevektor V gegeben.
Eine Drehung ist durch das Drehzentrum Z und den Drehwinkel w gegeben.
Eine Spiegelung ist durch die Spiegelachse PQ gegeben.
Eine Streckung ist durch das Streckzentrum Z und den Streckfaktor k gegeben.

Die Urfigur aller Abbildungen ist ein Dreieck ABC, dessen Koordinaten beliebig verändert werden können. Die Zielfigur wird grafisch dargestellt und auch ihre Koordinaten werden berechnet.

```
<!DOCTYPE html>
<html>
<head>
<title>Geometrische Abbildungen</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.00">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Schiebung Drehung Spiegelung Streckung">

<style>
body  {background-color:#E8E8F8; color:black; font-family:Arial;
       font-size:19px; font-weight:normal; text-align:left; padding-left:2%;}
p     {font-size:12px;}
.cd1  {border: 1px solid #666666; background-color: #FFFFE8; font-size: 15px;
       font-weight: bold; margin: 4px;}
.cd2  {border: 2px solid #666666; background-color: #EED8D8; font-size: 16px;
       font-weight: bold; margin: 4px;}
#MyCanvas {border: 1px solid black;}
fieldset  {border: 2px solid black; width: 200px; }
</style>
```

```
<script src="graph.js"></script>

<script>

function smart() {
// Anpassung an Smartphones
   var smallScr = false;
   var smallWidth = 800;
   var scrWidth = window.screen.width;
   if (scrWidth <= smallWidth) { smallScr = true; }
   if (smallScr) {
      document.querySelector('meta[name="viewport"]').setAttribute('content','initial-scale=0.80');
      let end = document.body.offsetHeight;
      document.getElementById("dc").style.top = end + 60 + "px";
      document.getElementById("dc").style.left = 20 + "px";
      liste = document.getElementsByClassName("cd2");
      for(var i = 0; i < liste.length; i++) { liste[i].style.fontSize = "24px"; }
   }
}

var childWindow;
var child = false;
window.addEventListener( "unload", function() { closeChild(); } );

function closeChild() {
// Hilfetext schließen
     if (!childWindow || childWindow.closed) { return; }
     if (child) { childWindow.document.close(); childWindow.close(); }
}

function infoText() {
   // Hilfetext anzeigen
    if (child) { closeChild(); child = false; return; }
    child = true;
    childWindow = window.open('','childWindow','top=20, left=480, width=580, height=720, srollbars=yes, menubar=no, toolbar=no');
    childWindow.document.open();
    childWindow.document.write('<html><head><meta name="viewport" content="width=device-width, initial-scale=1.0"></head>');
    childWindow.document.write('<body style="background-color:#F8F8E8; font-family: sans-serif; font-size:14px; text-size-adjust:none;
                          padding-left:10px; padding-bottom:10px;">');
    childWindow.document.write('<br><input type="button" value="Print" onclick="print();"><br><br>');
    childWindow.document.write('<b>Geometrische Abbildungen</b><br><br>');
    childWindow.document.write('Unter einer Abbildung versteht man die Erzeugung einer Bildfigur aus<br>');
    childWindow.document.write('einer gegebenen Urfigur entsprechend einer festgelegten Vorschrift.<br>');
    childWindow.document.write('Die Zuordnung von Bildfigur zur Urfigur muss eindeutig sein.<br><br>');
    childWindow.document.write('Zwei geometrische Figuren heißen <b>kongruent</b> (deckungsgleich), wenn<br>');
    childWindow.document.write('sie in Form und Fläche übereinstimmen. Dann sind die zugeordneten<br>');
    childWindow.document.write('Winkel gleich groß und die zugeordneten Strecken sind gleich lang.<br><br>');
    childWindow.document.write('Abbildungen, bei welchen Bildfigur und Urfigur deckungsgleich sind,<br>');
    childWindow.document.write('heißen <b>Kongruenzabbildungen</b>. Diese werden durch Spiegelungen<br>');
    childWindow.document.write('und Bewegungen (Schiebungen und Drehungen) realisiert.<br><br>');
    childWindow.document.write('Zwei geometrische Figuren heißen <b>ähnlich</b> (formgleich), wenn sie<br>');
    childWindow.document.write('in ihrer Form übereinstimmen, d.h. die zugeordneten Winkel und die<br>');
    childWindow.document.write('zugeordneten Streckenverhältnisse sind gleich groß.<br><br>');
    childWindow.document.write('Abbildungen, bei welchen die Bildfigur die gleiche Form hat wie die<br>');
    childWindow.document.write('Urfigur, heißen <b>Ähnlichkeitsabbildungen</b>. Diese werden durch so<br>');
    childWindow.document.write('genannte zentrische Streckungen realisiert.<br><br>');
    childWindow.document.write('<b>(1) Schiebung:</b> Verschiebe jeden Punkt in die gleiche Richtung und um die<br>');
    childWindow.document.write('gleiche Länge. Eine Schiebung wird durch einen Pfeil (Vektor) festgelegt.<br><br>');
    childWindow.document.write('<b>(2) Drehung:</b> Drehe jeden Punkt in die gleiche Richtung um den gleichen<br>');
    childWindow.document.write('Winkel um einen festen Punkt (Drehzentrum). Eine Drehung ist durch das<br>');
    childWindow.document.write('Drehzentrum und den Drehwinkel festgelegt.<br><br>');
    childWindow.document.write('<b>(3) Spiegelung:</b> Spiegle jeden Punkt an einer Spiegelachse. Bildpunkt<br>');
    childWindow.document.write('und Urpunkt liegen auf einer Geraden, welche normal zur Achse verläuft.<br>');
    childWindow.document.write('Dabei liegt der Bildpunkt auf der anderen Seite der Achse genauso weit<br>');
    childWindow.document.write('entfernt von der Achse wie der Urpunkt.<br><br>');
    childWindow.document.write('<b>(4) Streckung:</b> Urpunkt und Bildpunkt liegen auf dem gleichen Strahl durch<br>');
    childWindow.document.write('das Streckzentrum. Dabei ist der Abstand des Bildpunktes vom Zentrum<br>');
    childWindow.document.write('k-Mal so groß, wie der Abstand des Urpunktes vom Zentrum. Dabei wird<br>');
    childWindow.document.write('der Faktor k als Streckfaktor bezeichnet.<br>');
    childWindow.document.write('</body></html>');
}

kw = 10;
var A = new TPoint;
var B = new TPoint;
var C = new TPoint;
var D = new TPoint;
var E = new TPoint;
var F = new TPoint;
var M = new TPoint;
var P = new TPoint;
var Q = new TPoint;
var V = new TPoint;
var Z = new TPoint;

// Standardwerte der Abbildungen
var wert01 = '-6,-3';     // Schiebevektor V.x, V.y
var wert02 = '-2,1,90';   // Drehzentrum Z.x, Z.y, Drehwinkel w
var wert03 = '2,0,-5,7';  // Zwei Punkte der Spiegelachse P.x, P.y, Q.x, Q.y
var wert04 = '7,9,2';     // Streckzentrum Z.x, Z.y, Streckfaktor k

// Werte der Abbildungen
var wert1, wert2, wert3, wert4;
```

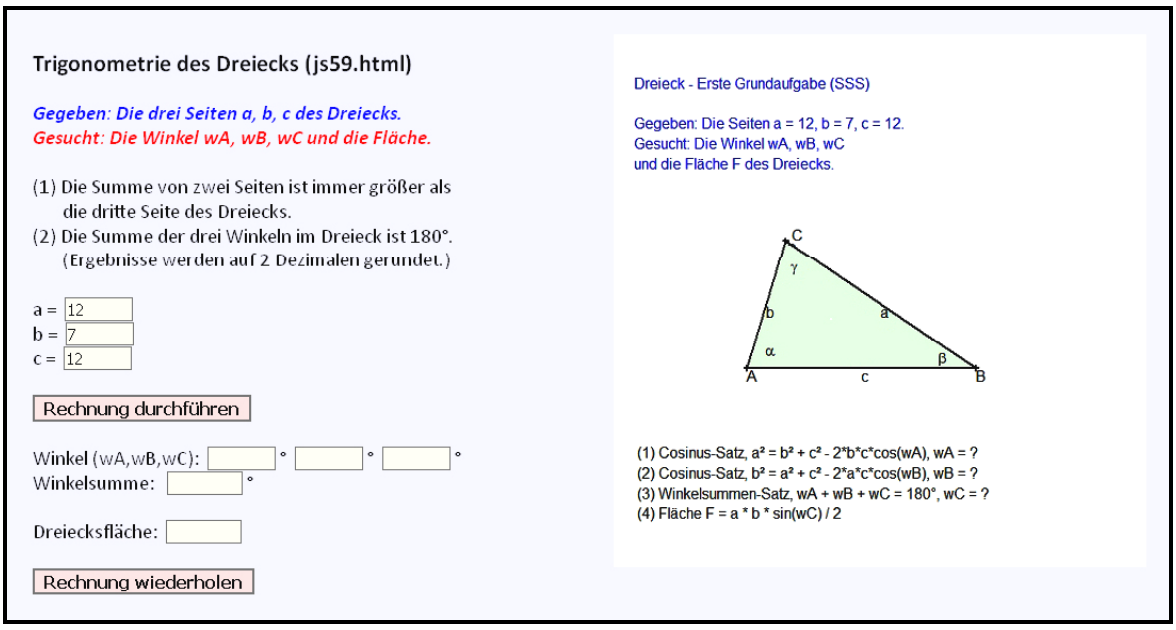
```
function start() {
// Daten erfassen
   kw = 1 * document.MyForm.ein0.value;
   if ((kw < 10) | (kw > 20)) {
     kw = 10;
     document.MyForm.ein0.value = kw;
   }
   initCanvas(kb,2,1)
   A.x = 1 * document.MyForm.ein1.value;
   A.y = 1 * document.MyForm.ein2.value;
   B.x = 1 * document.MyForm.ein3.value;
   B.y = 1 * document.MyForm.ein4.value;
   C.x = 1 * document.MyForm.ein5.value;
   C.y = 1 * document.MyForm.ein6.value;
   fillColor('rgba(255,0,0,0.25)');
   drawLine3(A,B,C);
   drawPoint('A',A);
   drawPoint('B',B);
   drawPoint('C',C);
}

function schieb() {
// Schiebung
   start();
   var text = 'Schiebung: V.X, V.Y';
   wert1 = '-6,-3';
   wert1 = prompt(text,wert1);
   var arr = wert1.split(',');
   V.x = 1 * arr[0].trim();
   V.y = 1 * arr[1].trim();

   D.x = A.x + V.x;
   D.y = A.y + V.y;
   E.x = B.x + V.x;
   E.y = B.y + V.y;
   F.x = C.x + V.x;
   F.y = C.y + V.y;

   fillColor('rgba(0,0,255,0.25)');
   drawLine3(D,E,F);

   drawWidth(1);
   drawColor('rgb(255,0,0)');
   drawLine(A,D);
   drawLine(B,E);
   drawLine(C,F);

   drawColor('rgb(0,0,0)');
   drawWidth(1);
   drawPoint('D',D);
   drawPoint('E',E);
   drawPoint('F',F);
   drawPosText(-kw+1,kw-2,'Schiebungen');
   drawPosText(-kw+1,-kw+3,'Schiebepfeil = (' + V.x + '/' + V.y + ')');
   drawPosText(-kw+1,-kw+2,'Urdreieck: A('+A.x+'/'+A.y+'), B('+B.x+'/'+B.y+'), C('+C.x+'/'+C.y+')');
   drawPosText(-kw+1,-kw+1,'Bilddreieck: D('+D.x+'/'+D.y+'), E('+E.x+'/'+E.y+'),
                                         F('+F.x+'/'+F.y+')');
}

function dreh() {
// Drehung
   start();
   var text = 'Drehung: Zentrum.X, Zentrum.Y, Drehwinkel';
   wert2 = '-2,1,90';
   wert2 = prompt(text,wert2);
   var arr = wert2.split(',');
   Z.x = 1 * arr[0].trim();
   Z.y = 1 * arr[1].trim();
   var w = 1 * arr[2].trim();
   var w0 = w*Math.PI/180;

   drawPoint('Z',Z);
   var ra = Math.sqrt((A.x-Z.x)*(A.x-Z.x) + (A.y-Z.y)*(A.y-Z.y));
   var wa = Math.atan((A.y-Z.y) / (A.x-Z.x));
   wa = 180*wa/Math.PI;
   fillArc(Z,ra,wa,w+wa);

   var rb = Math.sqrt((B.x-Z.x)*(B.x-Z.x) + (B.y-Z.y)*(B.y-Z.y));
   var wb = Math.atan((B.y-Z.y) / (B.x-Z.x));
   wb = 180*wb/Math.PI;
   fillArc(Z,rb,wb,w+wb);

   var rc = Math.sqrt((C.x-Z.x)*(C.x-Z.x) + (C.y-Z.y)*(C.y-Z.y));
   var wc = Math.atan((C.y-Z.y) / (C.x-Z.x));
   wc = 180*wc/Math.PI;
   fillArc(Z,rc,wc,w+wc);
```

```
   D.x = Z.x + (A.x-Z.x)*Math.cos(w0) - (A.y-Z.y)*Math.sin(w0);
   D.y = Z.y + (A.x-Z.x)*Math.sin(w0) + (A.y-Z.y)*Math.cos(w0);
   drawPoint('D',D);

   E.x = Z.x + (B.x-Z.x)*Math.cos(w0) - (B.y-Z.y)*Math.sin(w0);
   E.y = Z.y + (B.x-Z.x)*Math.sin(w0) + (B.y-Z.y)*Math.cos(w0);
   drawPoint('E',E);

   F.x = Z.x + (C.x-Z.x)*Math.cos(w0) - (C.y-Z.y)*Math.sin(w0);
   F.y = Z.y + (C.x-Z.x)*Math.sin(w0) + (C.y-Z.y)*Math.cos(w0);
   drawPoint('F',F);

   fillColor('rgba(0,0,255,0.25)');
   drawLine3(D,E,F);

   drawWidth(2);
   drawColor('rgb(255,0,0)');
   drawLine(D,E);
   drawLine(E,F);
   drawLine(F,D);

   drawColor('rgb(0,0,0)');
   drawWidth(1);
   drawLine(A,B);
   drawLine(B,C);
   drawLine(C,A);
   w = round2(w);
   D.x = round2(D.x);
   D.y = round2(D.y);
   E.x = round2(E.x);
   E.y = round2(E.y);
   F.x = round2(F.x);
   F.y = round2(F.y);
   drawPoint('A',A);
   drawPoint('B',B);
   drawPoint('C',C);
   drawPoint('D',D);
   drawPoint('E',E);
   drawPoint('F',F);
   drawPoint('Z',Z);
   drawPosText(-kw+1,kw-2,'Drehungen');
   drawPosText(-kw+1,-kw+3,'Drehzentrum Z = (' + Z.x + '/' + Z.y + '), Drehwinkel w = ' + w + '°');
   drawPosText(-kw+1,-kw+2,'Urdreieck: A('+A.x+'/'+A.y+'), B('+B.x+'/'+B.y+'), C('+C.x+'/'+C.y+')');
   drawPosText(-kw+1,-kw+1,'Bilddreieck: D('+D.x+'/'+D.y+'), E('+E.x+'/'+E.y+'),
                                         F('+F.x+'/'+F.y+')');
}

function spiegel() {
// Spiegelung
   start();
   var text = 'Spiegelachse PQ: P.X, P.Y, Q.X, Q.Y';
   var wert3 = '2,0,-5,7';
   wert3 = prompt(text,wert3);
   var arr = wert3.split(',');
   P.x = 1 * arr[0].trim();
   P.y = 1 * arr[1].trim();
   Q.x = 1 * arr[2].trim();
   Q.y = 1 * arr[3].trim();

   drawWidth(2);
   drawLine1(P,Q);
   drawWidth(1);
   drawPoint('P',P);
   drawPoint('Q',Q);

   var w = 2 * Math.atan((Q.y-P.y) / (Q.x-P.x));

   D.x = P.x + (A.x-P.x)*Math.cos(w) + (A.y-P.y)*Math.sin(w);
   D.y = P.y + (A.x-P.x)*Math.sin(w) - (A.y-P.y)*Math.cos(w);
   drawPoint('D',D);

   E.x = P.x + (B.x-P.x)*Math.cos(w) + (B.y-P.y)*Math.sin(w);
   E.y = P.y + (B.x-P.x)*Math.sin(w) - (B.y-P.y)*Math.cos(w);
   drawPoint('E',E);

   F.x = P.x + (C.x-P.x)*Math.cos(w) + (C.y-P.y)*Math.sin(w);
   F.y = P.y + (C.x-P.x)*Math.sin(w) - (C.y-P.y)*Math.cos(w);
   drawPoint('F',F);

   fillColor('rgba(0,0,255,0.25)');
   drawLine3(D,E,F);

   drawWidth(2);
   drawColor('rgb(255,0,0)');
   drawLine(D,E);
   drawLine(E,F);
   drawLine(F,D);
```

```
   drawWidth(1);
   drawColor('rgb(255,0,0)');
   drawLine(A,D);
   drawLine(B,E);
   drawLine(C,F);

   drawColor('rgb(0,0,0)');
   drawPoint('D',D);
   drawPoint('E',E);
   drawPoint('F',F);

   D.x = round2(D.x);
   D.y = round2(D.y);
   E.x = round2(E.x);
   E.y = round2(E.y);
   F.x = round2(F.x);
   F.y = round2(F.y);
   drawPosText(-kw+1,kw-2,'Spiegelungen');
   drawPosText(-kw+1,-kw+3,'Spiegelachse durch P('+P.x+'/'+P.y+') und Q('+Q.x+'/'+Q.y+')');
   drawPosText(-kw+1,-kw+2,'Urdreieck: A('+A.x+'/'+A.y+'), B('+B.x+'/'+B.y+'), C('+C.x+'/'+C.y+')');
   drawPosText(-kw+1,-kw+1,'Bilddreieck: D('+D.x+'/'+D.y+'), E('+E.x+'/'+E.y+'),
                                         F('+F.x+'/'+F.y+')');
}

function streck() {
// Streckung
   start();
   var text = 'Streckung: Zentrum.X, Zentrum.Y, Streckfaktor';
   wert4 = '7,9,2';
   wert4 = prompt(text,wert4);
   var arr = wert4.split(',');
   Z.x = 1 * arr[0].trim();
   Z.y = 1 * arr[1].trim();
   var k = 1 * arr[2].trim();
   drawPoint('Z',Z);

   D.x = Z.x + k*(A.x - Z.x);
   D.y = Z.y + k*(A.y - Z.y);
   drawPoint('D',D);

   E.x = Z.x + k*(B.x - Z.x);
   E.y = Z.y + k*(B.y - Z.y);
   drawPoint('E',E);

   F.x = Z.x + k*(C.x - Z.x);
   F.y = Z.y + k*(C.y - Z.y);
   drawPoint('F',F);

   fillColor('rgba(0,0,255,0.25)');
   drawLine3(D,E,F);

   drawWidth(2);
   drawColor('rgb(255,0,0)');
   drawLine(D,E);
   drawLine(E,F);
   drawLine(F,D);

   drawWidth(1);
   drawColor('rgb(255,0,0)');
   drawLine(Z,A);
   drawLine(Z,B);
   drawLine(Z,C);
   drawLine(Z,D);
   drawLine(Z,E);
   drawLine(Z,F);

   drawColor('rgb(0,0,0)');
   drawPoint('D',D);
   drawPoint('E',E);
   drawPoint('F',F);

   D.x = round2(D.x);
   D.y = round2(D.y);
   E.x = round2(E.x);
   E.y = round2(E.y);
   F.x = round2(F.x);
   F.y = round2(F.y);
   drawPosText(-kw+1,kw-2,'Streckungen');
   drawPosText(-kw+1,-kw+3,'Streckzentrum Z = (' + Z.x + '/' + Z.y + '), Streckfaktor k = ' + k);
   drawPosText(-kw+1,-kw+2,'Urdreieck: A('+A.x+'/'+A.y+'), B('+B.x+'/'+B.y+'), C('+C.x+'/'+C.y+')');
   drawPosText(-kw+1,-kw+1,'Bilddreieck: D('+D.x+'/'+D.y+'), E('+E.x+'/'+E.y+'),
                                         F('+F.x+'/'+F.y+')');
}
```

```
function saveFile() {
// Den Canvas in eine Grafikdatei speichern
   var canvas = document.getElementById('MyCanvas');
   var dataURL = canvas.toDataURL('image/jpeg',1);
   var ext = '.jpg';
   var gname = 'pict';
   gname = prompt('Dateiname ohne Extension (.jpg)',gname);
   gname = gname + ext;
   var link = document.createElement('a');
   link.download = gname;
   link.href = dataURL;
   document.body.appendChild(link);
   link.click();
   document.body.removeChild(link);
}
function resetForm() {
// Original-Formular wieder herstellen
   document.getElementById("MyForm").reset();
   start();
   wert1 = wert01;
   wert2 = wert02;
   wert3 = wert03;
   wert4 = wert04;
}
</script>
</head>

<body onload = "initCanvas(kb,2,1); start();">
<div id='dc' style="position:absolute; top:50px; left:500px; ">
  <canvas id='MyCanvas' width='500' height='500' onmousedown = "showPos(event)">
     Your browser does not support HTML5 Canvas.
  </canvas>
</div>
<h3>Geometrische Abbildungen</h3>
<form id="MyForm" name="MyForm">
Koordinatensystem (10 ... 20):<br>
<input type="text" name="ein0" value="10" size="3" class="cd1">
<br>
Eingabe eines Dreiecks (ABC):<br>
A.x <input type="text" name="ein1" value="0" size="3" class="cd1"> 
A.y <input type="text" name="ein2" value="4" size="3" class="cd1">
<br>
B.x <input type="text" name="ein3" value="4" size="3" class="cd1"> 
B.y <input type="text" name="ein4" value="4" size="3" class="cd1">
<br>
C.x <input type="text" name="ein5" value="0" size="3" class="cd1"> 
C.y <input type="text" name="ein6" value="7" size="3" class="cd1">
<br>
<input type="button" name="Start" value="Start" class="cd2" onclick="start()"><br>
<br>
<fieldset>
<input type="button" name="Schiebung" value="Schiebung" class="cd2" onclick="schieb()"><br>
<input type="button" name="Drehung" value="Drehung" class="cd2" onclick="dreh()"><br>
<input type="button" name="Spiegelung" value="Spiegelung" class="cd2" onclick="spiegel()"><br>
<input type="button" name="Streckung" value="Streckung" class="cd2" onclick="streck()"><br>
<br>
<input type="button" name="save" value="SaveGraph" class="cd2" onclick="saveFile()"><br>
<input type="button" name="info" value="Info" class="cd2" onclick="infoText()"><br>
<input type="button" name="info" value="Reset" class="cd2" onclick="resetForm()"><br>
</fieldset>
<p>&copy Herbert Paukert</p>
</form>
<script>
   function showPos(ev) {
   // Punktkoodinaten anzeigen mit Mausklick
      var cRect = document.getElementById("dc").getBoundingClientRect();
      var cTop  = cRect.top;
      var cLeft = cRect.left;
      var x1 = (ev.clientX - cLeft);
      var y1 = (ev.clientY - cTop);
      var x0 = x1*2*kw/kb - kw;
      var y0 = (kb - y1)*2*kw/kb - kw;
      var info = round2(x0) + '  /  ' + round2(y0);
      alert(info);
   }
   document.MyForm.ein1.focus();
</script>
</body>
<script> smart(); </script>
</html>
```

[3.3.3] Trigonometrie des Dreiecks (js59.html)

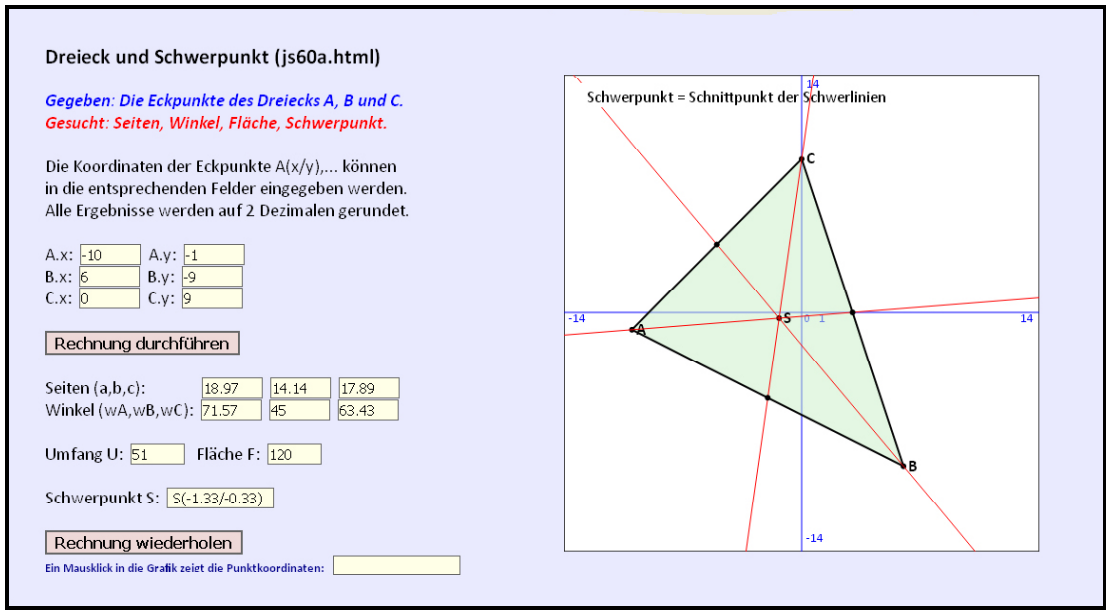



```
<!DOCTYPE html>
<html>
<head>
<title> JS59 </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Dreieck">

<style>

body {background-color:#EAEAFF; color:black; font-family:Calibri,Arial;
      font-size:19px; font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
.cd1 {border: 1px solid #666666; background-color: #FFFFE8; font-size: 15px;}
.cd2 {border: 2px solid #666666; background-color: #EED8D8; font-size: 16px; font-weight: bold;}

</style>

<script src="graph.js"></script>

<script>

   neu = false; // Programmflag

   function round2(x) { return Math.round(100*x)/100 }

   function deg(x) {
   // Umwandlung von Bogenmaß (x) in Gradmaß (y)
      var y;
      y = 180 * x / Math.PI;
      return y;
   }

   function flaeche(a,b,c) {
   // Fläche eines Dreiecks mit Seiten a, b, c
      var u,s,d,f;
      u = (1*a + 1*b + 1*c);
      s = u / 2;
      d = s*(s-a)*(s-b)*(s-c);
      if (d <= 0) { f = 0; }
      else { f = Math.sqrt(d); }
      return f.toFixed(2); }
   }
```

```
function winkel(a,b,c) {
// Winkel w zwischen Seite a und Seite b
   var x,y,z;
   x = (a*a + b*b - c*c) / (2*a*b);
   y = Math.acos(x);
   z = deg(y);
   return z.toFixed(2);
}


function Pruef() {

   var a = 1 * document.MyForm.ein1.value;
   var b = 1 * document.MyForm.ein2.value;
   var c = 1 * document.MyForm.ein3.value;

   var wa, wb, wc, f, h, x;

   f = flaeche(a,b,c);
   if (f <= 0) {
     alert('Die Seiten bilden KEIN Dreieck !');
     initCanvas(kb,2,0);
     return;
   }

   var A = new TPoint;
   var B = new TPoint;
   var C = new TPoint;

   h = 2*f/c;
   x = Math.sqrt(b*b - h*h);
   kw = 12;
   v = kw/2;

   A.x = 0-v;
   A.y = 0-v;
   B.x = c-v;
   B.y = 0-v;
   C.x = x-v;
   C.y = h-v;

   kw1 = Amax10(A.x,A.y,B.x,B.y,C.x,C.y);
   kw1 = Math.round(Math.abs(kw1) + 2);
   if (kw1 > 12) { kw = kw1; }
   else { kw = 12; }

   initCanvas(kb,2,0);
   drawColor("#000000");
   drawWidth(2);
   fillColor("rgba(200,240,200,0.5)");
   drawLine3(A,B,C);
   drawPoint("A",A);
   drawPoint("B",B);
   drawPoint("C",C);
   drawText(35,"Gegeben: Seiten a, b und c.");
   drawText(60,"Gesucht: Winkel wA, wB, wC und die Fläche F.");

   if (neu == true) { neu = false; return; }

   wa = winkel(b,c,a);
   wb = winkel(a,c,b);
   wc = winkel(a,b,c);
   wsum = 1*wa + 1*wb + 1*wc;

   document.MyForm.aus1.value = round2(wa);
   document.MyForm.aus2.value = round2(wb);
   document.MyForm.aus3.value = round2(wc);
   document.MyForm.aus4.value = round2(wsum);
   document.MyForm.aus5.value = round2(f);
}
```

```
function Zufall() {
   document.MyForm.reset();
   z = 4 + 16 * Math.random();
   p1 = Math.round(z);
   z = 4 + 16 * Math.random();
   p2 = Math.round(z);
   z = 4 + 16 * Math.random();
   p3 = Math.round(z);
   document.MyForm.ein1.value = p1;
   document.MyForm.ein2.value = p2;
   document.MyForm.ein3.value = p3;
   document.MyForm.aus1.value = '';
   document.MyForm.aus2.value = ' ';
   document.MyForm.aus3.value = ' ';
   document.MyForm.aus4.value = ' ';
   document.MyForm.aus5.value = ' ';
   neu = true;
   kw = 12;
   Pruef();
}

</script>
</head>

<body onload = "drawImg('dreieck1.jpg');">

<div id='dc' style="position:absolute; top:60px; left:600px ">
     <canvas id='MyCanvas' width='500' height='500'>
     Your browser does not support HTML5 Canvas.
  </canvas>
</div>

<form name="MyForm">
<br>
<h3>Trigonometrie des Dreiecks (js59.html)</h3>
<b><i><font color = "blue">
Gegeben: Die drei Seiten a, b, c des Dreiecks.<br>
</font></i></b>
<b><i><font color = "red">
Gesucht: Die Winkel wA, wB, wC und die Fläche.<br>
</font></i></b>
<br>
(1) Die Summe von zwei Seiten ist immer größer als<br>
       die dritte Seite des Dreiecks.<br>

(2) Die Summe der drei Winkeln im Dreieck ist 180°.<br>
       (Ergebnisse werden auf 2 Dezimalen gerundet.)<br>
<br>

a =  <input type="text" name="ein1" value="12" size="5" class="cd1"><br>
b =  <input type="text" name="ein2" value="7" size="5" class="cd1"><br>
c =  <input type="text" name="ein3" value="12" size="5" class="cd1"><br>
<br>

<input type="button" name="rech" value="Rechnung durchführen" class="cd2" onclick="Pruef()">
<br><br>

Winkel (wA,wB,wC):  <input type=text" name="aus1" value=" " size="5" class="cd1"> ° 
<input type=text" name="aus2" value=" " size="5" class="cd1"> ° 
<input type=text" name="aus3" value=" " size="5" class="cd1"> °
<br>

Winkelsumme:   <input type=text" name="aus4" value=" " size="6" class="cd1"> °<br>
<br>
Dreiecksfläche:  <input type=text" name="aus5" value=" " size="6" class="cd1"><br>
<br>
<input type="button" name="wied" value="Rechnung wiederholen" class="cd2" onclick="Zufall();">
<br>
</form>
</body>
</html>
```

[3.3.4] Dreieck und Schwerpunkt (js60a.html)




```
<!DOCTYPE html>
<html>
<head>
<title> JS60 </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Dreieck,Schwerpunkt">
<meta name="keywords" content=" ">

<style type = "text/css">
body {background-color:#EAEAFF; color:black; font-family:Calibri,Arial;
      font-size:19px; font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
.cd1 {border: 1px solid #666666; background-color: #FFFFE8; font-size: 15px;}
.cd2 {border: 2px solid #666666; background-color: #EED8D8; font-size: 16px; font-weight: bold;}
</style>

<script src="graph.js"></script>

<script>
neu = false;  // Programmflag
function round2(x) { return Math.round(100*x)/100 }

function Shap(P,Q) {
   // Streckenhalbierungspunkt
   H2 = new TPoint;
   H2.x = (P.x + Q.x) / 2;
   H2.y = (P.y + Q.y) / 2;
   return H2;
}
function Spunkt(P,Q,R) {
   // Schwerpunkt
   H2 = new TPoint;
   H2.x = (P.x + Q.x + R.x) / 3;
   H2.y = (P.y + Q.y + R.y) / 3;
   return H2;
}
function Slen(P,Q) {
  // Distanz zweier Punkte
  var z = (P.x - Q.x)*(P.x - Q.x) + (P.y - Q.y)*(P.y - Q.y);
  return Math.sqrt(z);
}
function Umfang(P,Q,R) {
  // Umfang eines Dreiecks
  var a1,b1,c1,u1;
  a1 = Slen(P,Q);
  b1 = Slen(Q,R);
  c1 = Slen(R,P);
  u1 = (a1 + b1 + c1);
  return u1;
}
```

```
function Flaeche(P,Q,R) {
  // Fläche eines Dreiecks
  var a1,b1,c1,u1,s1,f;
  a1 = Slen(P,Q);
  b1 = Slen(Q,R);
  c1 = Slen(R,P);
  u1 = (a1 + b1 + c1) / 2;
  s1 = u1*(u1-a1)*(u1-b1)*(u1-c1);
  if (s1 <= 0) { f = 0; }
  else { f = Math.sqrt(s1); }
  return f.toFixed(2); }
}

function Winkel(P,Q,R) {
  // Winkel w bei Punkt Q
  var a1,b1,c1,x1,y1,z1;
  a1 = Slen(Q,R);
  b1 = Slen(P,R);
  c1 = Slen(P,Q);
  x1 = (a1*a1 + c1*c1 - b1*b1) / (2*a1*c1);
  y1 = Math.acos(x1);
  z1 = deg(y1);
  return z1;
}

function Pruef() {

var A = new TPoint;
var B = new TPoint;
var C = new TPoint;
var D = new TPoint;
var E = new TPoint;
var F = new TPoint;
var S = new TPoint;
var T = new TPoint;

A.x = 1 * document.MyForm.ein1.value;
A.y = 1 * document.MyForm.ein2.value;
B.x = 1 * document.MyForm.ein3.value;
B.y = 1 * document.MyForm.ein4.value;
C.x = 1 * document.MyForm.ein5.value;
C.y = 1 * document.MyForm.ein6.value;

kw1 = Amax10(A.x,A.y,B.x,B.y,C.x,C.y);
kw1 = Math.round(Math.abs(kw1) + 4);
if (kw1 > 12) { kw = kw1; }
else { kw = 12; }

var sa, sb, sc, wa, wb, wc, s, u, f;
var info;

sa = Slen(B,C);
sb = Slen(A,C);
sc = Slen(A,B);
u  = Umfang(A,B,C);
f  = Flaeche(A,B,C);
wa = Winkel(B,A,C);
wb = Winkel(A,B,C);
wc = Winkel(A,C,B);

u  = round2(u);
f  = round2(f);
sa = round2(sa);
sb = round2(sb);
sc = round2(sc);
wa = round2(wa);
wb = round2(wb);
wc = round2(wc);

F = Shap(A,B)
D = Shap(A,C);
E = Shap(B,C)

S = Spunkt(A,B,C);
S.x = round2(S.x);
S.y = round2(S.y);

info = ' S(' + S.x + '/' + S.y + ')';
```

```
initCanvas(kb,2,1);
drawColor("#000000");
drawWidth(2);
fillColor("rgba(200,240,200,0.5)");
drawLine3(A,B,C);
drawPoint("A",A);
drawPoint("B",B);
drawPoint("C",C);

if (f == 0) {
document.MyForm.aus1.value = ' ';
document.MyForm.aus2.value = ' ';
document.MyForm.aus3.value = ' ';
document.MyForm.aus4.value = ' ';
document.MyForm.aus5.value = ' ';
document.MyForm.aus6.value = ' ';
document.MyForm.aus7.value = ' ';
document.MyForm.aus8.value = ' ';
document.MyForm.aus9.value = ' ';
initCanvas(kb,2);
alert('Die Punkte bilden kein Dreieck !');
return;
}

drawWidth(1);
drawColor("#FF0000");
drawPoint("S",S);
drawLine1(A,S);
drawLine1(B,S);
drawLine1(C,S);
drawPoint("",D);
drawPoint("",E);
drawPoint("",F);

drawText(30," Schwerpunkt = Schnittpunkt der Schwerlinien");
if (neu == true) { neu = false; return; }

document.MyForm.aus1.value = sa;
document.MyForm.aus2.value = sb;
document.MyForm.aus3.value = sc;
document.MyForm.aus4.value = wa;
document.MyForm.aus5.value = wb;
document.MyForm.aus6.value = wc;
document.MyForm.aus7.value = u;
document.MyForm.aus8.value = f;
document.MyForm.aus9.value = info;
}

function Zufall() {
// document.MyForm.reset();
z = Math.random(); if (z < 0.5) { v = -1;} else {v = 1; }
z = 1 + 9 * Math.random();
p1 = v * Math.round(z);
z = Math.random(); if (z < 0.5) { v = -1;} else {v = 1; }
z = 1 + 9 * Math.random();
p2 = v * Math.round(z);
z = Math.random(); if (z < 0.5) { v = -1;} else {v = 1; }
z = 1 + 9 * Math.random();
p3 = v * Math.round(z);
z = Math.random(); if (z < 0.5) { v = -1;} else {v = 1; }
z = 1 + 9 * Math.random();
p4 = v * Math.round(z);
z = Math.random(); if (z < 0.5) { v = -1;} else {v = 1; }
z = 1 + 9 * Math.random();
p5 = v * Math.round(z);
z = Math.random(); if (z < 0.5) { v = -1;} else {v = 1; }
z = 1 + 9 * Math.random();
p6 = v * Math.round(z);

document.MyForm.ein1.value = p1;
document.MyForm.ein2.value = p2;
document.MyForm.ein3.value = p3;
document.MyForm.ein4.value = p4;
document.MyForm.ein5.value = p5;
document.MyForm.ein6.value = p6;
```

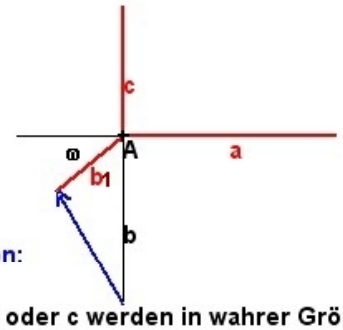
```
document.MyForm.aus1.value = ' ';
document.MyForm.aus1.value = ' ';
document.MyForm.aus2.value = ' ';
document.MyForm.aus3.value = ' ';
document.MyForm.aus4.value = ' ';
document.MyForm.aus5.value = ' ';
document.MyForm.aus6.value = ' ';
document.MyForm.aus7.value = ' ';
document.MyForm.aus8.value = ' ';
document.MyForm.aus9.value = ' ';
document.MyForm.aus10.value = ' ';
neu = true;
kw = 12;
Pruef();
}
</script>
</head>

<body onload = "drawImg('dreieck2.jpg');">
<div id='dc' style="position:absolute; top:60px; left:600px ">
     <canvas id='MyCanvas' width='500' height='500' onmousedown = "showPos(event)" >
     Your browser does not support HTML5 Canvas.
  </canvas>
</div>
<form name="MyForm">
<h3>Dreieck und Schwerpunkt (js60.html)</h3>
<b><i><font color = "blue">
Gegeben: Die Eckpunkte des Dreiecks A, B und C.<br>
</font></i></b>
<b><i><font color = "red">
Gesucht: Seiten, Winkel, Fläche, Schwerpunkt.<br>
</font></i></b>
<br>
Die Koordinaten der Eckpunkte A(x/y),... können<br>
in die entsprechenden Felder eingegeben werden.<br>
Alle Ergebnisse werden auf 2 Dezimalen gerundet.<br>
<br>
A.x:  <input type="text" name="ein1" value="-10" size="5" class="cd1"> 
A.y:  <input type="text" name="ein2" value="-1" size="5" class="cd1">
<br>
B.x:  <input type="text" name="ein3" value="6" size="5" class="cd1"> 
B.y:  <input type="text" name="ein4" value="-9" size="5" class="cd1">
<br>
C.x:  <input type="text" name="ein5" value="0" size="5" class="cd1"> 
C.y:  <input type="text" name="ein6" value="9" size="5" class="cd1">
<br><br>
<input type="button" name="rech" value="Rechnung durchführen" class="cd2" onclick="Pruef();">
<br><br>
Seiten (a,b,c):             
<input type=text" name="aus1" value=" " size="5" class="cd1"> 
<input type=text" name="aus2" value=" " size="5" class="cd1"> 
<input type=text" name="aus3" value=" " size="5" class="cd1"><br>
Winkel (wA,wB,wC):  <input type=text" name="aus4" value=" " size="5" class="cd1"> 
<input type=text" name="aus5" value=" " size="5" class="cd1"> 
<input type=text" name="aus6" value=" " size="5" class="cd1"><br>
<br>
Umfang U:  <input type=text" name="aus7" value=" " size="4" class="cd1">  
Fläche F:  <input type=text" name="aus8" value=" " size="4" class="cd1">
<br><br>
Schwerpunkt S:  <input type=text" name="aus9" value=" " size="12" class="cd1">
<br><br>
<input type="button" name="wied" value="Rechnung wiederholen" class="cd2" onclick="Zufall();">
<br>
<b><font size="-1" color="darkblue">
Ein Mausklick in die Grafik zeigt die Punktkoordinaten:  
<input type="text" name="aus10" value=" " size="15" class="cd1">
</font></b>
<br>
</form>
<script>
   function showPos(ev) {
   var x1 = ev.clientX - 600;
   var y1 = ev.clientY - 60;
   var x0 = x1*2*kw/kb - kw;
   var y0 = (kb - y1)*2*kw/kb - kw;
   var info = round2(x0) + '  /  ' + round2(y0);
   document.MyForm.aus10.value = info;
   }
</script>
</body>
</html>
```

[3.4] Dreidimensionale Darstellungen

Der Schrägriss

Der Schrägriss ist ein Verfahren mit dem räumliche Körper in der Zeichenebene abgebildet werden.

Betrachten wir die linke untere Ecke eines Quaders, dann stehen die drei Kanten a, b, c aufeinander paarweise senkrecht. Es sind beispielsweise a = 5 cm, b = 4 cm und c = 3 cm.

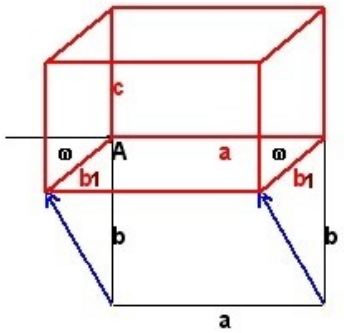


Schrägriss-Konstruktion:

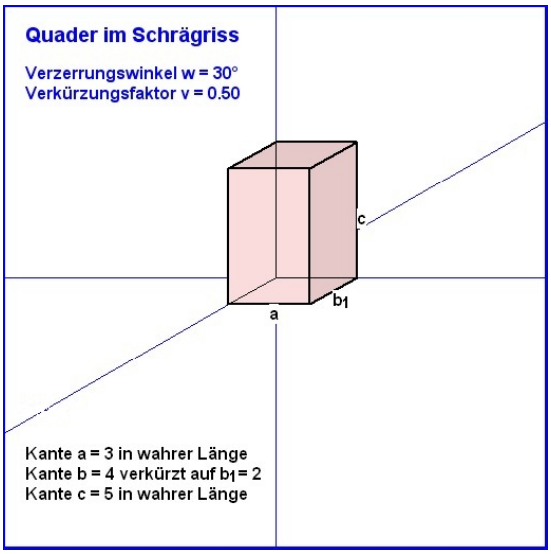
[1] Kanten parallel zu a oder c werden in wahrer Größe abgebildet.
[2] Kanten parallel zu b sind verkürzt und um einen Winkel geneigt. Ihre Bildstrecken b_1 in der Zeichenebene schließen mit der Kante a den Winkel w ein und werden mit einem Faktor v (v < 1) multipliziert. Der Pfeil symbolisiert, wie die wahre Kante b auf die verkürzte und geneigte Kante b_1 in der Zeichnung abgebildet wird.

Wir wollen nun den Quader mit Kanten a = 5 cm, b = 4 cm, c = 3 cm schrittweise im Schrägriss zeichnen:

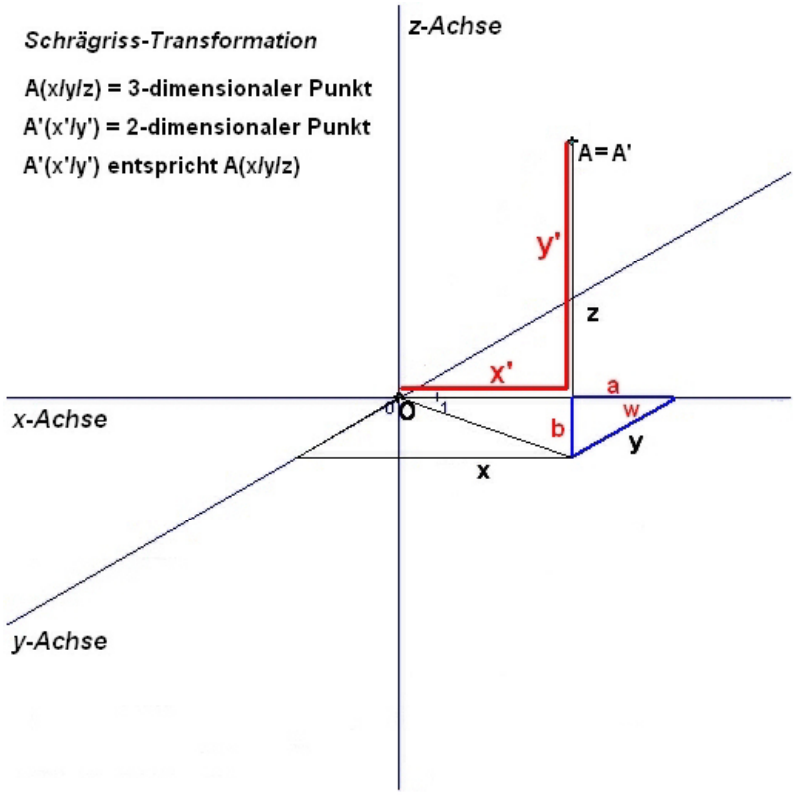
(1) Das Basisrechteck mit a und b wird in wahrer Größe gezeichnet.
(2) Die Bildkanten b_1 der Kanten b schließen mit Kante a einen Verzerrungswinkel w = 30 ° ein und werden halbiert. Verkürzungsfaktor ist somit v = 1 / 2 und es gilt daher $b_1 = b / 2$.



(3) Das Basisrechteck wird im Schrägriss fertig gezeichnet. Dabei entsteht ein Parallelogramm.
(4) Nun werden die vier Höhenkanten c in wahrer Länge gezeichnet.
(5) Zuletzt wird die Deckfläche gezeichnet und der Quader ist fertig.



Herleitung der Transformationsformel für die Schrägriss-Abbildung mit **k** = Schrägrissverkürzung (v) und **w** = Schrägrisswinkel (w).



x‘ = x – a
y‘ = z – b

a = y * cos(w) bzw. a = k*y * cos(w), weil y im Schrägriss verkürzt erscheint.
b = y * sin(w) bzw. b = k*y * sin(w), weil y im Schrägriss verkürzt erscheint.

x‘ = x – k*y * cos(w)
y‘ = z – k*y * sin(w)

Wenn der Schrägrisswinkel w im Gradmaß angegeben ist, dann muss er ins Bogenmaß (z) umgerechnet werden (z = rad(w) = w * ¶ / 180). Damit ergibt sich die engültige Transformation.

Schrägriss-Transformation: P(x/y/z) → P(x‘/y‘)
k = Schrägrissverkürzung (srk)
w = Schrägrisswinkel (srw)

x‘ = x – k*y * cos(rad(w))
y‘ = z – k*y * sin(rad(w))

Führt man in Javascript für zwei- und dreidimensionale Punkte zwei unterschiedliche Objekte ***TPoint*** und ***TPoint3*** ein, dann gilt:

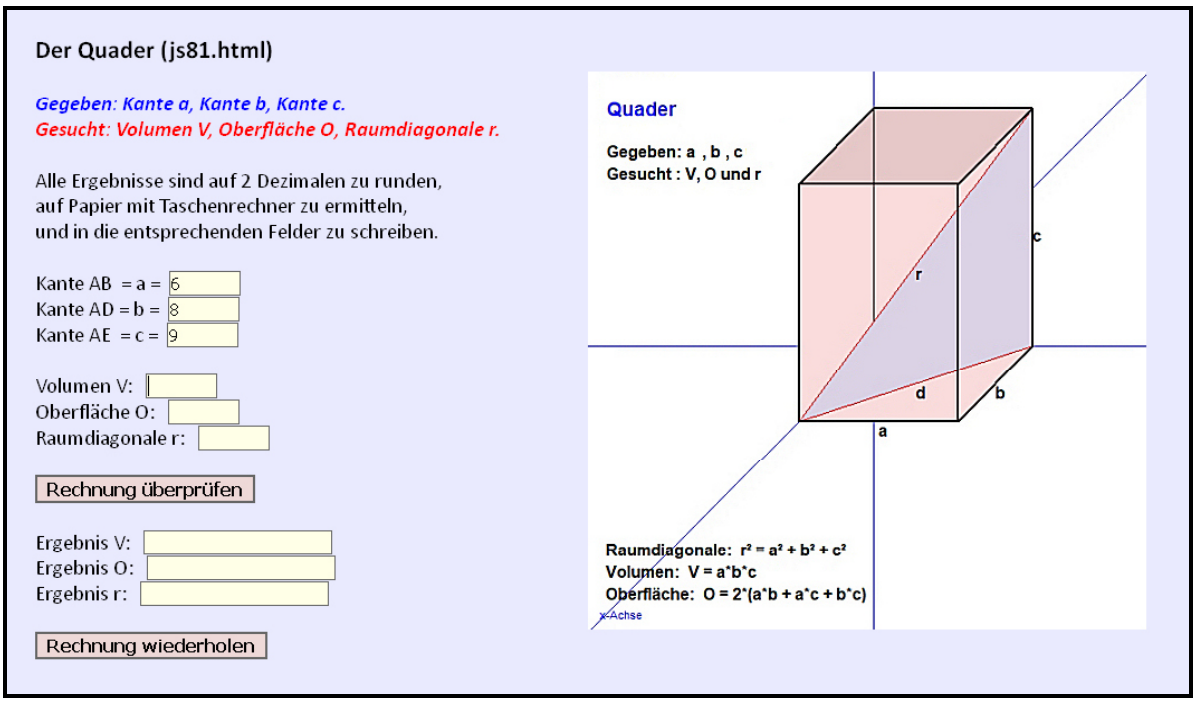
```
var srk = 0.5;   // Schrägrissverkürzung
var srw = 30;    // Schrägrisswinkel

function TPoint(tx, ty) {
  // Konstruktor für das Punkt-Objekt (zweidimensional)
  this.x = tx;
  this.y = ty;
  this.len = function() {
    var z = Math.sqrt(this.x*this.x + this.y*this.y);
    return z.toFixed(2);
  }
}

function TPoint3(tx, ty, tz) {
  // Konstruktor für das Punkt-Objekt (dreidimensional)
  this.x = tx;
  this.y = ty;
  this.z = tz;
  this.len = function() {
    var z = Math.sqrt(this.x*this.x + this.y*this.y + this.z*this.z);
    return z.tofixed(2);
  }
}

var P3 = new TPoint3;
P3.x = 5;  // beispielsweise
P3.y = 4;
P3.z = 3;

function Schraeg(P3) {
    // Schrägrisstransformation
    Q = new TPoint;
    Q.x = P3.x - srk * P3.y * Math.cos(rad(srw));
    Q.y = P3.z - srk * P3.y * Math.sin(rad(srw));
    return Q;
}

var P = new TPoint;
P = Schraeg(P3);  // das ergibt dann P(3.26/2)
```

Im folgenden Programm „**js81.html**“ ist ein Quader mit den Seiten a, b und c gegeben. Damit können die dreidimensionalen Koordinaten der acht Eckpunkte A3, B3, C3, D3, und E3, F3, G3, H3 sofort ermittelt werden: A3(0/0/0), B3(a/0/0), C3(a/b/0), D3(0/b/0) und E3(0/0/c), F3(a/0/c), G3(a/b/c), H3(0/b/c). Diese werden mit obiger Transformation in acht Punkte A, B, C, D, E, F, G mit zweidimensionalen Koordinaten umgewandelt. Dann können diese im Canvas gezeichnet und mit Strecken verbunden werden. Auf diese einfache Weise erhält man ein schönes Schrägrissbild des Quaders.

Im Programm „**js85.html**“ ist ein Zylinder mit dem Radius r und der Höhe h gegeben. Die dreidimensionalen Koordinaten des Mittelpunktes *M3(u/v/0)* der Grundfläche und jene des Mittelpunktes *S3(u/v/h)* der Deckfläche werden mittels *M = Schraeg(M3)* und *S = Schraeg(S3)* in die zweidimensionalen Punkte *M* und *S* transformiert. Der Einfachheit wegen wird mit *u = 0* und *v = 0* der Punkt *M3* in den Koordinatenursprung gelegt. Dann werden die beiden Kreise $k_1(M,r)$ und $k_2(S,r)$ im Schrägriss als zwei Ellipsen $ell_1(M,r,srk*r)$ und $ell_2(S,r,srk*r)$ dargestellt. Dabei sind *r* die Hauptachse und *srk*r* die verkürzte Nebenachse. Zusätzlich wird noch ein Schnitt-Rechteck ABCD gezeichnet.

In die Programme ist die JavaScript-Datei „**graph.js**“ eingebunden. Sie enthält alle wichtigen Grafikroutinen, die dann von den Programmen aufgerufen werden.

[3.4.1] Der Quader (js81.html)

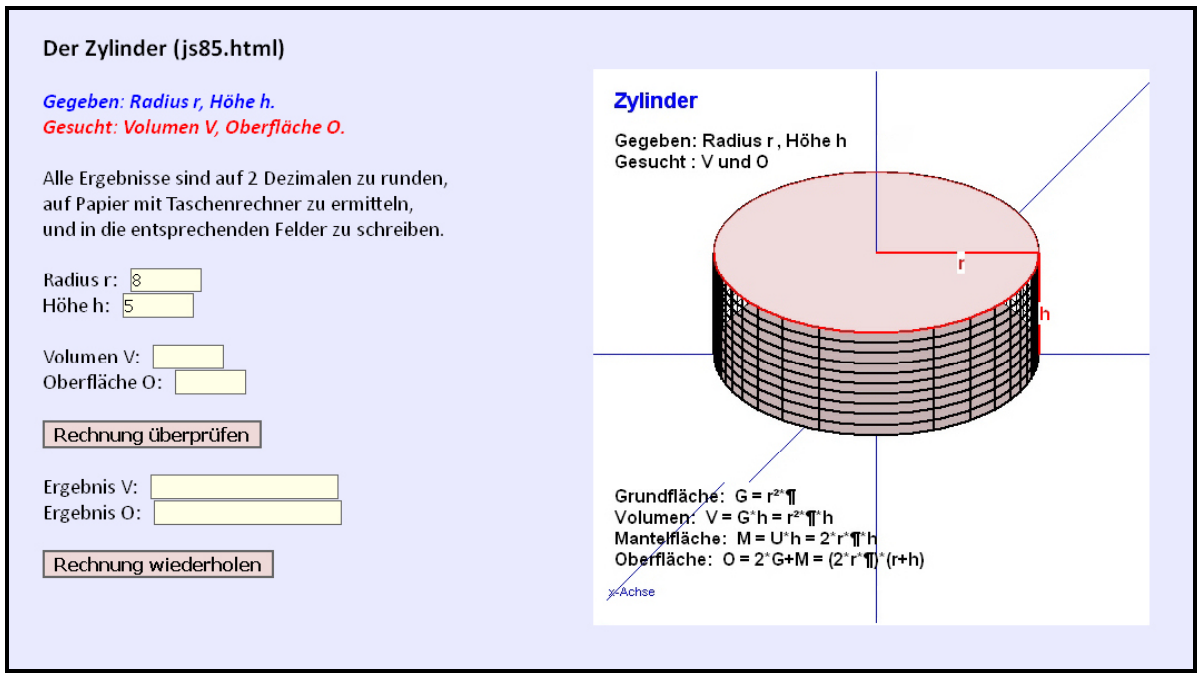



```
<!DOCTYPE html>
<html>
<head>
<title> JS81, Quader </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Quader">
<meta name="keywords" content=" ">

<style type = "text/css">
body {background-color:#EAEAFF; color:black; font-family:Calibri,Arial; font-size:19px; font-
weight:normal; text-align:left; margin:2%; padding-left:2%;}
.cd1 {border: 1px solid #666666; background-color: #FFFFE8; font-size: 15px;}
.cd2 {border: 2px solid #666666; background-color: #EED8D8; font-size: 16px; font-weight: bold;}
</style>

<script src = "graph.js"></script>

<script>
  var neu = false;  // Programmflag
  var p1, p2, p3;   // Aufgaben-Parameter

function round2(x) { return Math.round(100*x)/100 }

function Pruef() {
  // Ausrechnung und Zeichnung
  kw = 12;
  p1 = Math.abs(document.MyForm.ein1.value);
  p2 = Math.abs(document.MyForm.ein2.value);
  p3 = Math.abs(document.MyForm.ein3.value);
  kw1 = Max6(p1,p2,p3) + 2;
  if (kw1 > kw) { kw = kw1; }
  initCanvas(kb,2,0);

  var a = p1;
  var b = p2;
  var c = p3;

  A3 = new TPoint3;
  B3 = new TPoint3;
  C3 = new TPoint3;
  D3 = new TPoint3;
  E3 = new TPoint3;
  F3 = new TPoint3;
  G3 = new TPoint3;
  H3 = new TPoint3;
```

```
A = new TPoint;
B = new TPoint;
C = new TPoint;
D = new TPoint;
E = new TPoint;
F = new TPoint;
G = new TPoint;
H = new TPoint;

A3.x = 0;
A3.y = 0;
A3.z = 0;

B3.x = a;
B3.y = 0;
B3.z = 0;

C3.x = a;
C3.y = b;
C3.z = 0;

D3.x = 0;
D3.y = b;
D3.z = 0;

E3.x = 0;
E3.y = 0;
E3.z = c;

F3.x = a;
F3.y = 0;
F3.z = c;

G3.x = a;
G3.y = b;
G3.z = c;

H3.x = 0;
H3.y = b;
H3.z = c;

A = Schraeg(A3);
B = Schraeg(B3);
C = Schraeg(C3);
D = Schraeg(D3);

E = Schraeg(E3);
F = Schraeg(F3);
G = Schraeg(G3);
H = Schraeg(H3);

drawWidth(2);
drawColor("#000000");

drawLine(A,B);
drawLine(B,C);
drawLine(C,D);
drawLine(D,A);

drawLine(E,F);
drawLine(F,G);
drawLine(G,H);
drawLine(H,E);

drawLine(A,E);
drawLine(B,F);
drawLine(C,G);
drawLine(D,H);

drawPoint("A",A);
drawPoint("B",B);
drawPoint("C",C);
drawPoint("D",D);

drawColor("#000000");
fillColor("rgba(180,240,180,0.5)");
drawLine4(D,C,G,H);
drawLine4(B,C,G,F);
drawLine4(E,F,G,H);
```

```
  drawPoint("E",E);
  drawPoint("F",F);
  drawPoint("G",G);
  drawPoint("H",H);

  drawWidth(1);
  fillColor("rgba(250,200,200,0.5");
  drawLine3(D,F,B);

  info = 'Quader: a = ' + a + ', b = ' + b + ', c = ' + c;
  drawText(30,info);

  if (neu ==true) { neu = false; return; }

  var volumen = a * b * c;
  var obflae  = 2 * (a*b + a*c + b*c);
  var diagon  = Math.sqrt(a*a + b*b + c*c);

  document.MyForm.aus1.value = volumen;
  document.MyForm.aus2.value = obflae;
  document.MyForm.aus3.value = round2(diagon);
}

function Zufall() {
  // Zufallszahlen
  z = 2 + 8 * Math.random();
  p1 = Math.round(z);
  document.MyForm.ein1.value = p1;
  z = 2 + 8 * Math.random();
  p2 = Math.round(z);
  document.MyForm.ein2.value = p2;
  z = 2 + 8 * Math.random();
  p3 = Math.round(z);
  document.MyForm.ein3.value = p3;
  document.MyForm.aus1.value = '';
  document.MyForm.aus2.value = '';
  document.MyForm.aus3.value = '';
  neu = true;
  Pruef();
}
</script>
</head>

<body onload = "drawImg('quader.jpg')">

<div id='dc' style="position:absolute; top:60px; left:550px ">
  <canvas id='MyCanvas' width='500' height='500'>
     Your browser does not support HTML5 Canvas.
  </canvas>
</div>

<form name="MyForm">
<br>
<h3>Der Quader (js81.html) </h3>
<b><i><font color = "blue">
Gegeben: Kante a, Kante b, Kante c.<br>
</font></i></b>
<b><i><font color = "red">
Gesucht: Volumen V, Oberfläche O, Raumdiagonale r.<br>
</font></i></b>
<br>
Alle Ergebnisse sind auf zwei Dezimalen zu runden.<br>
Rechnungen können mit Taschenrechner auf einem<br>
Blatt Papier ausgeführt und dann überprüft werden.<br>
<br>
Kante AB  = a =  <input type="text" name="ein1" value="6" size="5" class="cd1"><br>
Kante AD = b =  <input type="text" name="ein2" value="8" size="5" class="cd1"><br>
Kante AE  = c =  <input type="text" name="ein3" value="9" size="5" class="cd1"><br>
<br>
<input type="button" name="rech" value="Rechnung überprüfen" class="cd2" onclick="Pruef();"><br>
<br>
Volumen V:   <input type=text" name="aus1" value="" size="5" class="cd1"><br>
Oberfläche O:   <input type=text" name="aus2" value="" size="5" class="cd1"><br>
Raumdiagonale r:   <input type=text" name="aus3" value="" size="5" class="cd1"><br>
<br>
<input type="button" name="wied" value="Rechnung wiederholen" class="cd2" onclick="Zufall();">
<br>
</form>
</body>
</html>
```

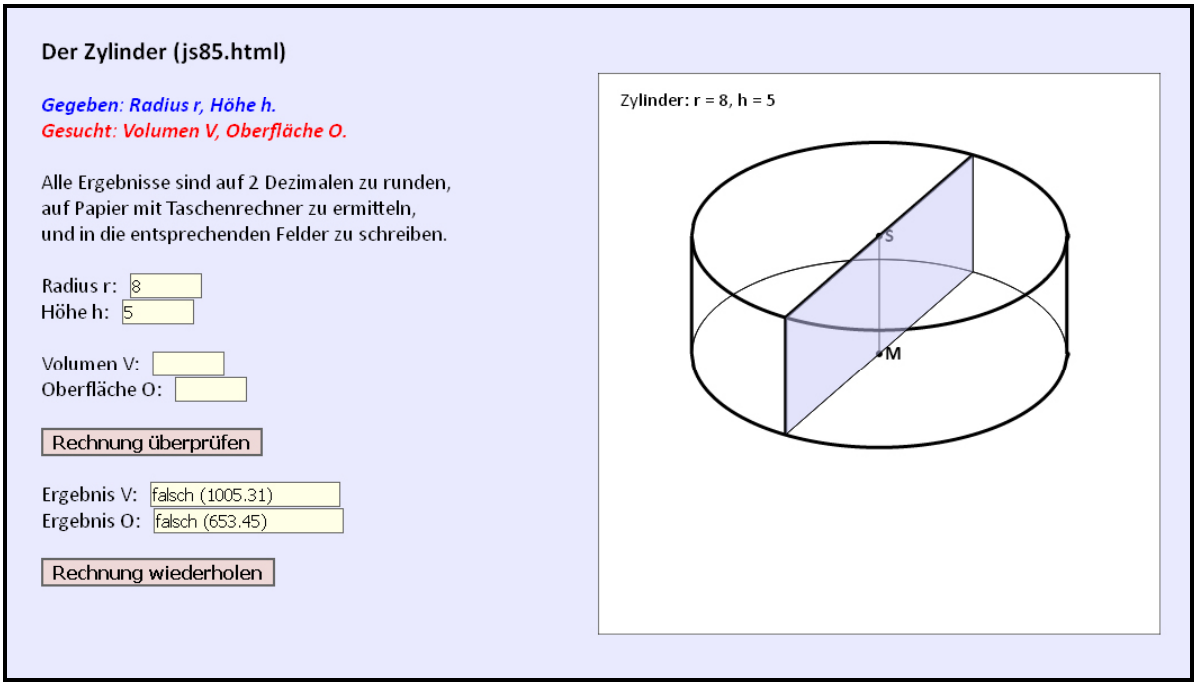
[3.4.2] Der Zylinder (js85.html)




```
<!DOCTYPE html>
<html>
<head>
<title> JS85, Zylinder </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Zylinder ">
<meta name="keywords" content=" ">

<style type = "text/css">
body {background-color:#EAEAFF; color:black; font-family:Calibri,Arial;
      font-size:19px; font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
.cd1 {border: 1px solid #666666; background-color: #FFFFE8; font-size: 15px;}
.cd2 {border: 2px solid #666666; background-color: #EED8D8; font-size: 16px; font-weight: bold;}
</style>

<script src = "graph.js"></script>

<script>
var neu = false;  // Programmflag
var p1, p2;       // Eingabe-Parameter

function round2(x) { return Math.round(100*x)/100 }

function Pruef() {
  kw = 12;
  p1 = Math.abs(document.MyForm.ein1.value);
  p2 = Math.abs(document.MyForm.ein2.value);
  kw1 = Max6(p1,p2) + 2;
  if (kw1 > kw) { kw = kw1; }

  initCanvas(kb,2,0);

  var r = p1;
  var h = p2;

  M3 = new TPoint3;
  S3 = new TPoint3;

  M = new TPoint;
  S = new TPoint;
  A = new TPoint;
  B = new TPoint;
  C = new TPoint;
  D = new TPoint;
  R = new TPoint;
  T = new TPoint;
  U = new TPoint;
  V = new TPoint;
```

```
  M3.x = 0;
  M3.y = 0;
  M3.z = 0;
  S3.x = 0;
  S3.y = 0;
  S3.z = h;

  M = Schraeg(M3);
  S = Schraeg(S3);
  A.x = -r;
  A.y = 0;
  B.x = r;
  B.y = 0;
  C.x = r;
  C.y = h;
  D.x = -r;
  D.y = h;

  var a = r;
  var b = r;

  drawWidth(2);
  drawColor("#000000");
  drawWidth(3);
  drawEll(M,a,srk*b,-1);
  drawEll(S,a,srk*b,1);
  drawWidth(1);
  drawEll(M,a,srk*b,1);
  drawWidth(3);
  drawEll(S,a,srk*b,-1);
  drawLine(B,C);
  drawLine(D,A);
  drawPoint("M",M)
  drawPoint("S",S)

  R.x = r/2;
  R.y = funEll(R.x,a,srk*b);
  T.x = -R.x;
  T.y = -funEll(T.x,a,srk*b);
  drawWidth(1);
  drawLine(R,T);
  drawWidth(3);
  U.x = r/2;
  U.y = funEll(U.x,a,srk*b) + h;
  V.x = -U.x;
  V.y = -funEll(V.x,a,srk*b) + h;
  drawLine(U,V);
  drawLine(T,V);
  drawWidth(1);
  drawLine(R,U)
  drawLine(M,S);
  fillColor("rgba(200,200,250,0.5)");
  drawLine4(R,T,V,U);
  info = 'Zylinder: r = ' + r + ', h = ' + h;
  drawText(30,info);
  if (neu ==true) { neu = false; return; }

  basis = r * h * Math.PI;
  mantel = 2 * r * h * Math.PI;
  volumen = basis * h;
  obflae = 2 * basis + mantel;
  document.MyForm.aus1.value = round2(volumen);
  document.MyForm.aus2.value = round2(obflae);
}

function Zufall() {
z = 3 + 7 * Math.random();
p1 = Math.round(z);
document.MyForm.ein1.value = p1;
z = 2 + 4 * Math.random();
p2 = Math.round(z);
document.MyForm.ein2.value = p2;
document.MyForm.aus1.value = '';
document.MyForm.aus2.value = '';
neu = true;
Pruef();
}
</script>
</head>
```

```
<body onload = "drawImg('zylinder.jpg')">

<div id='dc' style="position:absolute; top:60px; left:550px ">
  <canvas id='MyCanvas' width='500' height='500'>
     Your browser does not support HTML5 Canvas.
  </canvas>
</div>

<form name="MyForm">
<br>
<h3>Der Zylinder (js85.html)</h3>
<b><i><font color = "blue">
Gegeben: Radius r, Höhe h.<br>
</font></i></b>
<b><i><font color = "red">
Gesucht: Volumen V, Oberfläche O.<br>
</font></i></b>
<br>
Alle Ergebnisse sind auf zwei Dezimalen zu runden.<br>
Rechnungen können mit Taschenrechner auf einem<br>
Blatt Papier ausgeführt und dann überprüft werden.<br>
<br>
Radius r:   <input type="text" name="ein1" value="8" size="5" class="cd1"><br>
Höhe h:   <input type="text" name="ein2" value="5" size="5" class="cd1"><br>
<br>
<input type="button" name="rech" value="Rechnung überprüfen" class="cd2" onclick="Pruef()"><br>
<br>
Volumen V:   <input type=text" name="aus1" value="" size="5" class="cd1"><br>
Oberfläche O:   <input type=text" name="aus2" value="" size="5" class="cd1"><br>
<br>

<input type="button" name="wied" value="Rechnung wiederholen" class="cd2" onclick=Zufall()>
<br>
</form>
</body>
</html>
```



Schrägrissbild des Zylinders

[3.5] JavaScript-Datei „graph.js“

```
// ------------------------------------------------------------------------------
// "graph.js", Sammlung von Grafik-Routinen
// (c) Herbert Paukert, Version 4.0 am 10.12.2018
// ------------------------------------------------------------------------------

var kw = 10;     // halbe Weltbreite
var kb = 500;    // ganze Bildbreite

var visa = 1;    // Sichtbarkeit des Koordinatensystems
var srk = 0.5;   // Schrägrissverkürzung
var srw = 30;    // Schrägrisswinkel

function myTrim(x) { return x.replace(/^\s+|\s+$/gm,'') }
function round2(x) { return Math.round(100*x)/100 }
function round4(x) { return Math.round(10000*x)/10000 }
function deg(x) { return x * 180 / Math.PI; }
function rad(x) { return x * Math.PI / 180; }

function TPoint(tx, ty) {
  // Konstruktor für das Punkt-Objekt (zweidimensional)
  this.x = tx;
  this.y = ty;
  this.len = function() {
    var z;
    z = Math.sqrt(this.x*this.x + this.y*this.y);
    return round2(z);
  }
  this.win = function() {
    var z;
    if ((this.x < 0) && (this.y == 0)) { z = 180; return z;}
    if ((this.x == 0) && (this.y < 0)) { z = 270; return z; }
    if ((this.x == 0) && (this.y == 0)) { z = 0; return z; }
    z = Math.atan(this.y/this.x);
    if (z == Infinity) { z = 90; return z; }
    z = deg(Math.abs(z));
    if ((this.x < 0) && (this.y > 0)) { z = 180 - z; }
    if ((this.x < 0) && (this.y < 0)) { z = 180 + z; }
    if ((this.x > 0) && (this.y < 0)) { z = 360 - z; }
    if (z >= 360 ) {z = z - 360; }
    return round2(z);
  }
}

function TPoint3(tx, ty, tz) {
  // Konstruktor für das Punkt-Objekt (dreidimensional)
  this.x = tx;
  this.y = ty;
  this.z = tz;
  this.len = function() {
    var z = Math.sqrt(this.x*this.x + this.y*this.y + this.z*this.z);
    return round2(z);
  }
  this.win = function() {
    if ((this.x < 0) && (this.y == 0)) { z = 180; return z;}
    if ((this.x == 0) && (this.y < 0)) { z = 270; return z; }
    if ((this.x == 0) && (this.y == 0)) { z = 0; return z; }
    var z = Math.atan(this.y/this.x);
    if (z == Infinity) { z = 90; return z; }
    z = deg(Math.abs(z));
    if ((this.x < 0) && (this.y > 0)) { z = 180 - z; }
    if ((this.x < 0) && (this.y < 0)) { z = 180 + z; }
    if ((this.x > 0) && (this.y < 0)) { z = 360 - z; }
    if (z >= 360 ) {z = z - 360; }
    return round2(z);
  }
}

function Max6(a,b,c,d,e,f) {
    // Maximum von 6 Zahlen
    var z = a;
    if (b > z) { z = b; }
    if (c > z) { z = c; }
    if (d > z) { z = d; }
    if (e > z) { z = e; }
    if (f > z) { z = f; }
    return z;
}
```

```
function Min6(a,b,c,d,e,f) {
  // Minimum von 6 Zahlen
  var z = a;
  if (b < z) { z = b; }
  if (c < z) { z = c; }
  if (d < z) { z = d; }
  if (e < z) { z = e; }
  if (f < z) { z = f; }
  return z;
}
function Amax10(a,b,c,d,e,f,g,h,i,j) {
  // Absolutes Maximum von 10 Zahlen
  var z = Math.abs(a);
  if (Math.abs(b) > z) { z = Math.abs(b); }
  if (Math.abs(c) > z) { z = Math.abs(c); }
  if (Math.abs(d) > z) { z = Math.abs(d); }
  if (Math.abs(e) > z) { z = Math.abs(e); }
  if (Math.abs(f) > z) { z = Math.abs(f); }
  if (Math.abs(g) > z) { z = Math.abs(g); }
  if (Math.abs(h) > z) { z = Math.abs(h); }
  if (Math.abs(i) > z) { z = Math.abs(i); }
  if (Math.abs(j) > z) { z = Math.abs(j); }
  return z;
}
function Schraeg(P9) {
  // Schrägrisstransformation
  var P = new TPoint;
  P.x = P9.x - srk * Math.cos(rad(srw)) * P9.y;
  P.y = P9.z - srk * Math.sin(rad(srw)) * P9.y;
  return P;
}
function w2p(P9,kw,kb) {
  // World-To-Picture-Transformation von Weltpunkt P9 auf Bildpunkt Q
  var Q = new TPoint;
  Q.x = round2((P9.x + kw) * kb / (2*kw));
  Q.y = round2(kb - (P9.y + kw) * kb / (2*kw));
  return Q;
}
function p2w(P9,kw,kb) {
  // Picture-To-World-Transformation von Bildpunkt P9 auf Weltpunkt Q
  var Q = new TPoint;
  Q.x = round2(P9.x*2*kw/kb - kw);
  Q.y = round2((kb – P9.y)*2*kw/kb - kw);
  return Q;
}
function initCanvas(kb,n,visa) {
// Canvas initialisieren
  var canvas = document.getElementById('MyCanvas');
  var ctx = canvas.getContext('2d');
  ctx.font = '12pt Calibri';
  ctx.lineWidth = 1;
  ctx.fillStyle = "white";
  ctx.fillRect(0,0,kb,kb);
  ctx.strokeStyle = "black";
  ctx.strokeRect(0,0,kb,kb);
  if ((n == 2) && (visa == 1)) { drawCoord2(kw,kb); }
  if ((n == 3) && (visa == 1)) { drawCoord3(kw,kb); }
}
function clearImg(n,visa) {
// Grafik löschen
  initCanvas(kb,n,visa);
}
function drawImg(name) {
// Eine JPG-Grafikdatei laden
  var canvas = document.getElementById('MyCanvas');
  var ctx = canvas.getContext('2d');
  var img = new Image();
  img.src = name;
  img.onload = function() { ctx.drawImage(img,0,0); }
}
function drawCoord2(kw,kb) {
  // Koordinatensystem zeichnen (2-dimensional)
  var canvas = document.getElementById('MyCanvas');
  var ctx = canvas.getContext('2d');
  ctx.strokeStyle = "black";
  ctx.strokeRect(0,0,kb,kb);
  ctx.font = '10pt Calibri';
```

```
    ctx.beginPath();
    var g = kb / 2;
    ctx.lineWidth = 1;
    ctx.setLineDash([2,2]);
    ctx.strokeStyle = "blue";
    ctx.moveTo(0,g);
    ctx.lineTo(2*g,g);
    ctx.moveTo(g,0);
    ctx.lineTo(g,2*g);
    ctx.stroke();
    ctx.setLineDash([]);
    var h = g / kw;
    ctx.fillStyle = "blue";
    ctx.fillText('0',g+2,g+10);
    ctx.fillText('1',g+h,g+10);
    ctx.fillText(-kw,5,g+10);
    ctx.fillText(kw,2*g-20,g+10);
    ctx.fillText(kw,g+5,15);
    ctx.fillText(-kw,g+5,2*g-10);
    ctx.strokeStyle = "black";
    ctx.closePath();
  }

 function drawCoord3(kw,kb) {
    // Koordinatensystem zeichnen (3-dimensional)
    var canvas = document.getElementById('MyCanvas');
    var ctx = canvas.getContext('2d');
    ctx.strokeStyle = "black";
    ctx.strokeRect(0,0,kb,kb);
    ctx.font = '10pt Calibri';
    ctx.beginPath();
    var g = kb / 2;
    ctx.lineWidth = 1;
    ctx.strokeStyle = "blue";
    ctx.moveTo(0,g);
    ctx.lineTo(2*g,g);
    ctx.moveTo(g,0);
    ctx.lineTo(g,2*g);
    var y0 = Math.round(g * Math.tan(rad(srw)));
    var y1 = g + y0;
    var y2 = g - y0;
    ctx.moveTo(0,y1);
    ctx.lineTo(2*g,y2);
    ctx.stroke();
    var h = g / kw;
    ctx.fillStyle = "blue";
    ctx.fillText('0',g+2,g+10);
    ctx.fillText('1',g+h,g+10);
    ctx.fillText('x-Achse',5,g+10);
    ctx.fillText(kw,2*g-20,g+10);
    ctx.fillText(kw,g+5,15);
    ctx.fillText('z-Achse',g+5,2*g-10);
    ctx.fillText('y-Achse',0+5,y1-10);  // ctx.fillText(-kw,2*g-20,y2+10);
    ctx.strokeStyle = "black";
    ctx.closePath();
  }

  function drawPoint(tex,P9) {
    // Punkt P9 zeichnen und mit tex beschriften in Farbe schwarz
    var canvas = document.getElementById('MyCanvas');
    var ctx = canvas.getContext('2d');
    var P1 = new TPoint;
    P1 = w2p(P9,kw,kb);
    var r9 = 3;
    ctx.font = 'bold 13pt Calibri';
    ctx.beginPath();
    ctx.arc(P1.x,P1.y,r9,0,2*Math.PI);
    ctx.fillStyle = "black"; ctx.fill(); ctx.fillText(tex,P1.x+5,P1.y+5);
    ctx.closePath();
  }

  function killPoint(tex,P9) {
    // Punkt P9 zeichnen und mit tex beschriften in Farbe weiß
    var canvas = document.getElementById('MyCanvas');
    var ctx = canvas.getContext('2d');
    var P1 = new TPoint;
    P1 = w2p(P9,kw,kb);
    var r9 = 3;
    ctx.font = 'bold 13pt Calibri';
    ctx.beginPath();
    ctx.arc(P1.x,P1.y,r9,0,2*Math.PI);
    ctx.fillStyle = "white"; ctx.fill(); ctx.fillText(tex,P1.x+5,P1.y+5);
    ctx.closePath();
  }
```

```
function drawPosText(x,y,tex) {
// Text positioniert ausgeben (in Weltkoordinaten)
  var canvas = document.getElementById('MyCanvas');
  var ctx = canvas.getContext('2d');
  ctx.font = 'bold 13pt Calibri';
  var P1 = new TPoint;
  P1.x = x;
  P1.y = y;
  P1 = w2p(P1,kw,kb);
  ctx.fillStyle = "black";   // ctx.fillStyle = "red";
  ctx.fillText(tex,P1.x,P1.y);
}

function drawText(y,tex) {
// Text positioniert ausgeben (in Bildkoordinaten)
  var canvas = document.getElementById('MyCanvas');
  var ctx = canvas.getContext('2d');
  ctx.fillStyle = "white";
  ctx.fillRect(1,y-20,kb/3,25);
  ctx.font = '13pt Calibri';
  ctx.fillStyle = "black";
  ctx.fillText(tex,20,y);
}

function drawWidth(f) {
  // Liniendicke
  var canvas = document.getElementById('MyCanvas');
  var ctx = canvas.getContext('2d');
  ctx.lineWidth = f;
}

function drawColor(f) {
  // Linienfarbe
  var canvas = document.getElementById('MyCanvas');
  var ctx = canvas.getContext('2d');
  ctx.strokeStyle = f;
}

function fillColor(f) {
  // Füllfarbe
  var canvas = document.getElementById('MyCanvas');
  var ctx = canvas.getContext('2d');
  ctx.fillStyle = f;
}

function drawCircle(M9,r9) {
  // Kreis zeichnen
  var canvas = document.getElementById('MyCanvas');
  var ctx = canvas.getContext('2d');
  var M1 = new TPoint;
  var P1 = new TPoint;
  var r1;
  M1 = M9;
  M1 = w2p(M1,kw,kb);
  P1.x = r9;
  P1.y = 0;
  P1 = w2p(P1,kw,kb);
  r1 = Math.abs(P1.x - kb/2);
  ctx.beginPath();
  ctx.arc(M1.x,M1.y,r1,0,2*Math.PI);
  ctx.closePath();
  ctx.stroke();
}

function fillCircle(M9,r9) {
  // Kreis füllen
  var canvas = document.getElementById('MyCanvas');
  var ctx = canvas.getContext('2d');
  var M1 = new TPoint;
  var P1 = new TPoint;
  var r1;
  M1 = M9;
  M1 = w2p(M1,kw,kb);
  P1.x = r9;
  P1.y = 0;
  P1 = w2p(P1,kw,kb);
  r1 = Math.abs(P1.x - kb/2);
  ctx.beginPath();
  ctx.arc(M1.x,M1.y,r1,0,2*Math.PI);
  ctx.closePath();
  ctx.fill();
  ctx.stroke();
}
```

```
  function fillArc(M9,r9,w1,w2) {
    // Kreissektor füllen
    var canvas = document.getElementById('MyCanvas');
    var ctx = canvas.getContext('2d');
    var M1 = new TPoint;
    var P1 = new TPoint;
    var P2 = new TPoint;
    var P3 = new TPoint;
    var r1;
    var anf = w1 * Math.PI / 180;
    var end = w2 * Math.PI / 180;
    P1.x = r9;
    P1.y = 0;
    P2.x = r9*Math.cos(anf) + M9.x;
    P2.y = r9*Math.sin(anf) + M9.y;
    P3.x = r9*Math.cos(end) + M9.x;
    P3.y = r9*Math.sin(end) + M9.y;
    P2 = w2p(P2,kw,kb);
    P3 = w2p(P3,kw,kb);
    M1 = w2p(M9,kw,kb);
    P1 = w2p(P1,kw,kb);
    r1 = Math.abs(P1.x - kb/2);
    fillColor('rgba(250,200,200,0.5');
    ctx.beginPath();
    ctx.arc(M1.x,M1.y,r1,-end,-anf);
    ctx.moveTo(M1.x,M1.y);
    ctx.lineTo(P2.x,P2.y);
    ctx.moveTo(M1.x,M1.y);
    ctx.lineTo(P3.x,P3.y);
    ctx.closePath();
    ctx.fill();
    ctx.stroke();
  }

  function drawLine(R9,S9) {
    //  Strecke durch die Punkte R9 und S9 zeichnen
    var canvas = document.getElementById('MyCanvas');
    var ctx = canvas.getContext('2d');
    var U1 = new TPoint;
    var V1 = new TPoint;
    U1 = R9;
    V1 = S9;
    U1 = w2p(U1,kw,kb);
    V1 = w2p(V1,kw,kb);
    ctx.beginPath();
//  ctx.strokeStyle = "black"
    ctx.moveTo(U1.x,U1.y);
    ctx.lineTo(V1.x,V1.y);
    ctx.closePath();
    ctx.stroke();
  }

  function drawLine3(W7,W8,W9) {
    //  gefülltes Dreieck W7, W8, W9, W7 zeichnen
    var canvas = document.getElementById('MyCanvas');
    var ctx = canvas.getContext('2d');
    var V = new TPoint;
    ctx.beginPath();
    V = W7;
    V = w2p(V,kw,kb);
    ctx.moveTo(V.x,V.y);
    V = W8;
    V = w2p(V,kw,kb);
    ctx.lineTo(V.x,V.y);
    V = W9;
    V = w2p(V,kw,kb);
    ctx.lineTo(V.x,V.y);
    V = W7;
    V = w2p(V,kw,kb);
    ctx.lineTo(V.x,V.y);
    ctx.moveTo(V.x,V.y);
    ctx.closePath();
    ctx.fill();
    ctx.stroke();
  }
```

```
function drawLine4(W6,W7,W8,W9) {
  //  gefülltes Vierieck W6, W7, W8, W9, W6 zeichnen
  var canvas = document.getElementById('MyCanvas');
  var ctx = canvas.getContext('2d');
  var V = new TPoint;
  ctx.beginPath();
  V = W6;
  V = w2p(V,kw,kb);
  ctx.moveTo(V.x,V.y);
  V = W7;
  V = w2p(V,kw,kb);
  ctx.lineTo(V.x,V.y);
  V = W8;
  V = w2p(V,kw,kb);
  ctx.lineTo(V.x,V.y);
  V = W9;
  V = w2p(V,kw,kb);
  ctx.lineTo(V.x,V.y);
  V = W6;
  V = w2p(V,kw,kb);
  ctx.lineTo(V.x,V.y);
  ctx.moveTo(V.x,V.y);
  ctx.closePath();
  ctx.fill();
  ctx.stroke();
}

function drawLine1(R9,S9) {
  //  Gerade durch die Punkte R9 und S9 zeichnen
  var canvas = document.getElementById('MyCanvas');
  var ctx = canvas.getContext('2d');
  //  ctx.lineWidth = 1;
  //  ctx.strokeStyle = "black";
  var U = new TPoint;
  var V = new TPoint;
  U = R9;
  V = S9;
  var g,k1,d1;
  g = kb / 2;
  ctx.beginPath();
  U = w2p(U,kw,kb);
  V = w2p(V,kw,kb);
  if (U.x == V.x) {
     U.y = 0;
     V.y = 2*g;
     ctx.moveTo(U.x,U.y);
     ctx.lineTo(V.x,V.y);
     ctx.stroke();
     return;
  }
  if (U.y == V.y) {
     U.x = 0;
     V.x = 2*g;
     ctx.moveTo(U.x,U.y);
     ctx.lineTo(V.x,V.y);
     ctx.stroke();
     return;
  }
  k1 = (U.y - V.y) / (U.x - V.x);
  d1 = U.y - k1*U.x;
  U.x = 0; U.y = k1*U.x + d1;
  V.x = 2*g; V.y = k1*V.x + d1;
  ctx.moveTo(U.x,U.y);
  ctx.lineTo(V.x,V.y);
  ctx.closePath();
  ctx.stroke();
}

function funEll(x,a,b) {
   // Ellipsenfunktion
   var y;
   y = (b/a) * Math.sqrt(a*a - x*x);
   if (Math.abs(x) > a) {y = 0;}
   return y;
}
```

```
function drawEll(M9,a,b,r9) {
    // Ellipsen zeichnen mit Mittelpunkt M9(x/y) und mit r9 = +/-1
    var canvas = document.getElementById('MyCanvas');
    var ctx = canvas.getContext('2d');
    var U = new TPoint;
    var V = new TPoint;
    var x,xd,x1,x2,y1,y2,xMin,xMax;
    xMin = -a;
    xMax = a;
    x = xMin;
    xd = 2*kw/kb;
    x = x - xd;
    ctx.beginPath();

    do {
       x = x + xd;
       x1 = x;
       x2 = x + 2*xd;
       y1 = r9*funEll(x1,a,b);
       y2 = r9*funEll(x2,a,b);
       U.x = x1 + M9.x;
       U.y = y1 + M9.y;
       V.x = x2 + M9.x;
       V.y = y2 + M9.y;
       U = w2p(U,kw,kb);
       V = w2p(V,kw,kb);
       ctx.moveTo(U.x,U.y);
       ctx.lineTo(V.x,V.y);
    } while (x <= xMax);
    ctx.closePath();
    ctx.stroke();
}

function fillEll(M9,a,b,r9) {
    // Ellipsen füllen mit Mittelpunkt M9(x/y) und mit r9 = +/-1
    var canvas = document.getElementById('MyCanvas');
    var ctx = canvas.getContext('2d');
    var U = new TPoint;
    var V = new TPoint;
    var U0 = new TPoint;
    var V0 = new TPoint;
    var x,xd,x1,x2,y1,y2,xMin,xMax;
    xMin = -a;
    xMax = a;
    x = xMin;
    xd = 2*kw/kb;
    x = x - xd;
    ctx.beginPath();

    do {
       x = x + xd;
       x1 = x;
       x2 = x + 2*xd;
       y1 = r9*funEll(x1,a,b);
       y2 = r9*funEll(x2,a,b);
       U.x = x1 + M9.x;
       U.y = y1 + M9.y;
       U0.x = x1 + M9.x;
       U0.y = 0;
       U = w2p(U,kw,kb);
       U0 = w2p(U0,kw,kb);
       V.x = x2 + M9.x;
       V.y = y2 + M9.y;
       V0.x = x2 + M9.x;
       V0.y = 0;
       V = w2p(V,kw,kb);
       V0 = w2p(V0,kw,kb);
       ctx.beginPath();
       ctx.moveTo(U0.x,U0.y);
       ctx.moveTo(U.x,U.y);
       ctx.lineTo(V0.x,V0.y);
       ctx.lineTo(V.x,V.y);
       ctx.closePath();
       ctx.stroke();
    } while (x <= xMax);
}
```

```
// Funktions-Plotter OHNE "parser.js" -----------------------------------

var fpa, fpb, fpc, fpd;  // vier Funktionsparameter für 13 Funktionstypen
var fselect = 1;         // Funktions-Steuervariable (neu)
var fselect_old = 0;     // Funktions-Steuervariable (alt)

function fun01(x) { y = fpa*x + fpb; return y; }
function fun02(x) { y = fpa*x*x + fpb*x + fpc; return y; }
function fun03(x) { y = fpa*x*x*x + fpb*x*x + fpc*x + fpd; return y; }
function fun04(x) { y = 1/(x - fpa); return y; }
function fun05(x) { y = 1/((x - fpa)*(x - fpa)); return y; }
function fun06(x) { y = fpb * Math.sqrt(fpa*x); return y; }
function fun07(x) { y = (fpb/fpa) * Math.sqrt(fpa*fpa - x*x); return y; }
function fun08(x) { y = (fpb/fpa) * Math.sqrt(x*x - fpa*fpa); return y; }
function fun09(x) { y = fpa * Math.exp(fpb*x); return y; }
function fun10(x) { y = fpa * Math.log(fpb*x); return y; }
function fun11(x) { y = fpa * Math.sin(fpb*x + fpc); return y; }
function fun12(x) { y = fpa * Math.exp(fpb*x) * Math.sin(fpc*x); return y; }
function fun13(x) { y = fpa * Math.exp(-x*x/2) / Math.sqrt(2*Math.PI); return y; }

function selectParam() {
// Erzeugung der Funktions-Parameter
   fpa = 0; fpb = 0; fpc = 0; fpd = 0;
   if (fselect == 1) { fpa = 1; fpb = 1; }
   if (fselect == 2) { fpa = 1; }
   if (fselect == 3) { fpa = 0.5; fpb = -3; fpc = 4; }
   if (fselect == 4) { fpa = 3; }
   if (fselect == 5) { fpa = 3; }
   if (fselect == 6) { fpa = 1; fpb = 2; }
   if (fselect == 7) { fpa = 6; fpb = 4; }
   if (fselect == 8) { fpa = 2; fpb = 3 }
   if (fselect == 9) { fpa = 1; fpb = 1; }
   if (fselect == 10) { fpa = 1; fpb = 1; }
   if (fselect == 11) { fpa = 4; fpb = 1; }
   if (fselect == 12) { fpa = 1; fpb = -0.5; fpc = 4; }
   if (fselect == 13) { fpa = 8; fpb = 0; fpc = 0; }
}

function drawFunk(fselect) {
    // ausgewählte Funktionen zeichnen
    var canvas = document.getElementById('MyCanvas');
    var ctx = canvas.getContext('2d');
    var P9 = new TPoint;
    var Q9 = new TPoint;
    var xMin,xMax,x,y,xd,x1,x2,y1,y2;
    xMin = -kw; xMax = kw;
    x = xMin;
    xd = 2*kw/kb;
    x = x - xd;
    var kordinat = false;
    var asymptot = false;
    do {
       kordinat = false;
       asymptot = false;
       x = x + xd;
       x1 = x;
       x2 = x + 2*xd;
       if ((fselect < 1) || (fselect > 13)) { return; }
       if (fselect == 1)  { y1 = fun01(x1); y2 = fun01(x2); }
       if (fselect == 2)  { y1 = fun02(x1); y2 = fun02(x2); }
       if (fselect == 3)  { y1 = fun03(x1); y2 = fun03(x2); }
       if (fselect == 4)  { y1 = fun04(x1); y2 = fun04(x2); }
       if (fselect == 5)  { y1 = fun05(x1); y2 = fun05(x2); }
       if (fselect == 6)  { y1 = fun06(x1); y2 = fun06(x2); }
       if (fselect == 7)  { y1 = fun07(x1); y2 = fun07(x2); }
       if (fselect == 8)  { y1 = fun08(x1); y2 = fun08(x2); }
       if (fselect == 9)  { y1 = fun09(x1); y2 = fun09(x2); }
       if (fselect == 10) { y1 = fun10(x1); y2 = fun10(x2); }
       if (fselect == 11) { y1 = fun11(x1); y2 = fun11(x2); }
       if (fselect == 12) { y1 = fun12(x1); y2 = fun12(x2); }
       if (fselect == 13) { y1 = fun13(x1); y2 = fun13(x2); }
       if (y1 > 2*kw)    { y1 = kw; asymptot = true;  }
       if (y1 < (-2)*kw) { y1 = -kw; asymptot = true; }
       if (y2 > 2*kw)    { y2 = kw; asymptot = true;  }
       if (y2 < (-2)*kw) { y2 = -kw; asymptot = true; }
       if (Number.isNaN(y1)) { kordinat = true; y1 = 0; }
       if (Number.isNaN(y2)) { kordinat = true; y2 = 0; }
       P9.x = x1; P9.y = y1;
       Q9.x = x2; Q9.y = y2;
       P9 = w2p(P9,kw,kb);
       Q9 = w2p(Q9,kw,kb);
```

```
        if ( (Math.abs(y1) < 2*kw) && (Math.abs(y2) < 2*kw) ) {
           ctx.lineWidth = 2;
           ctx.beginPath();
           ctx.moveTo(P9.x,P9.y);
           ctx.lineTo(Q9.x,Q9.y);
           ctx.closePath();
           if (asymptot || (y1 == 0 && y2 ==0)) { ctx.strokeStyle = "white"; }
           else { ctx.strokeStyle = "red"; }
           ctx.stroke();
        }
      } while (x <= xMax);
    drawCoord2(kw,kb);
  }

// Funktions-Plotter MIT "parser.js" -------------------------------------

 function drawFun(kw,kb,s) {
    // kw = Halbbreite des Koordinatensystems, kb = Canvas-Bildbreite
    // Funktion fun(s,x) zeichnen, benötigt "parser.js"
    // s = Funktionsterm, x = Argument
    var canvas = document.getElementById('MyCanvas');
    var ctx = canvas.getContext('2d');
    var x,xd,x1,x2,y1,y2,xMin,xMax;
    ctx.font = '10pt Calibri';
    ctx.lineWidth = 2;
    ctx.strokeStyle = "red";
    var asymptot = false;
    var R = new TPoint;
    var S = new TPoint;
    xMin = -kw;
    xMax = kw;
    x = xMin;
    xd = 2*kw/kb;
    x = x - xd;

    do {
       asymptot = false;
       x = x + xd;
       x1 = x;
       y1 = fun(s,x1);
       x2 = x + 2*xd;
       y2 = fun(s,x2);
       if (y1 > 2*kw)    { y1 = kw;  asymptot = true; }
       if (y1 < (-2)*kw) { y1 = -kw; asymptot = true; }
       if (y2 > 2*kw)    { y2 = kw;  asymptot = true; }
       if (y2 < (-2)*kw) { y2 = -kw; asymptot = true; }
       if (Number.isNaN(y1)) { y1 = 0; }
       if (Number.isNaN(y2)) { y2 = 0; }
       R.x = x1;
       R.y = y1;
       R = w2p(R,kw,kb);
       S.x = x2;
       S.y = y2;
       S = w2p(S,kw,kb);
       if (asymptot || (y1 == 0 && y2 ==0)) { ctx.strokeStyle = "white"; }
       else { ctx.strokeStyle = "red"; }
       ctx.beginPath();
       ctx.moveTo(R.x,R.y);
       ctx.lineTo(S.x,S.y);
       ctx.stroke();
       ctx.closePath();
     } while (x <= xMax);
    drawCoord2(kw,kb);
  }

  function drawFun1(kw,kb,xa,xb,s) {
    // kw = Halbbreite des Koordinatensystems, kb = Canvas-Bildbreite
    // Funktion fun(s,x) zeichnen, zwischen xa und xb, benötigt "parser.js"
    // s = Funktionsterm, x = Argument
    var canvas = document.getElementById('MyCanvas');
    var ctx = canvas.getContext('2d');
    var x,xd,x1,x2,y1,y2,xMin,xMax,xx;
    ctx.font = '10pt Calibri';
    var asymptot = false;
    var R = new TPoint;
    var S = new TPoint;
    if (xa > xb) {xx = xa; xa = xb; xb = xx; }
    xMin = xa;
    xMax = xb;
    x = xMin;
    xd = 2*(xb-xa)/kb;
    x = x - xd;
```

```
    do {
       asymptot = false;
       x = x + xd;
       x1 = x;
       y1 = fun(s,x1);
       x2 = x + 2*xd;
       y2 = fun(s,x2);
       if (y1 > 2*kw)    { y1 = kw;  asymptot = true; }
       if (y1 < (-2)*kw) { y1 = -kw; asymptot = true; }
       if (y2 > 2*kw)    { y2 = kw;  asymptot = true; }
       if (y2 < (-2)*kw) { y2 = -kw; asymptot = true; }
       if (Number.isNaN(y1)) { y1 = 0; }
       if (Number.isNaN(y2)) { y2 = 0; }
       R.x = x1; R.y = y1;
       R = w2p(R,kw,kb);
       S.x = x2; S.y = y2;
       S = w2p(S,kw,kb);
       if (asymptot || (y1 == 0 && y2 ==0)) { ctx.strokeStyle = "white"; }
       else { ctx.strokeStyle = "red"; }
       ctx.beginPath();
       ctx.moveTo(R.x,R.y);
       ctx.lineTo(S.x,S.y);
       ctx.closePath();
       ctx.stroke();
     } while (x <= xMax);
    drawCoord2(kw,kb);
  }

  function fillFun1(kw,kb,xa,xb,s) {
   // kw = Halbbreite des Koordinatensystems, kb = Canvas-Bildbreite
    // Funktion fun(s,x) mit Farbe füllen, zwischen xa und xb, benötigt "parser.js"
    // s = Funktionsterm, x = Argument
    var canvas = document.getElementById('MyCanvas');
    var ctx = canvas.getContext('2d');
    var x,xd,x1,x2,y1,y2,xMin,xMax,xx;
    ctx.font = '10pt Calibri';
    var asymptot = false;
    var R = new TPoint;
    var S = new TPoint;
    var R0 = new TPoint;
    var S0 = new TPoint;
    if (xa > xb) {xx = xa; xa = xb; xb = xx; }
    xMin = xa;
    xMax = xb;
    x = xMin;
    xd = 2*(xb-xa)/kb;
    x = x - xd;
    do {
       asymptot = false;
       x = x + xd;
       x1 = x;
       y1 = fun(s,x1);
       x2 = x + 2*xd;
       y2 = fun(s,x2);
       if (y1 > 2*kw)    { y1 = kw;  asymptot = true; }
       if (y1 < (-2)*kw) { y1 = -kw; asymptot = true; }
       if (y2 > 2*kw)    { y2 = kw;  asymptot = true; }
       if (y2 < (-2)*kw) { y2 = -kw; asymptot = true; }
       if (Number.isNaN(y1)) { y1 = 0; }
       if (Number.isNaN(y2)) { y2 = 0; }
       R.x = x1;  R.y = y1;
       R0.x = x1; R0.y = 0;
       R = w2p(R,kw,kb);
       R0 = w2p(R0,kw,kb);
       S.x = x2;  S.y = y2;
       S0.x = x2; S0.y = 0;
       S = w2p(S,kw,kb);
       S0 = w2p(S0,kw,kb);
       if (asymptot || (y1 == 0 && y2 ==0)) { ctx.strokeStyle = "white"; }
       else { ctx.strokeStyle = "red"; }
       ctx.beginPath();
       ctx.moveTo(R0.x,R0.y);
       ctx.lineTo(R.x,R.y);
       ctx.moveTo(S0.x,S0.y);
       ctx.lineTo(S.x,S.y);
       ctx.closePath();
       ctx.stroke();
     } while (x <= xMax);
    drawCoord2(kw,kb);
  }
// ---------------------------------------------------------------------------
// Ende von "graph.js"
// ---------------------------------------------------------------------------
```

● Verzeichnis aller Bibliotheks-Funktionen von „graph.js“

```
var kw = 10;                          // halbe Weltbreite
var kb = 500;                         // ganze Bildbreite
var visa = 1;                         // Sichtbarkeit des Koordinatensystems
var srk = 0.5;                        // Schrägrissverkürzung
var srw = 30;                         // Schrägrisswinkel

function myTrim(x)                    // Führende und folgende Blanks von x entfernen
function round2(x)                    // x auf 2 Dezimalen runden
function round4(x)                    // x auf 4 Dezimalen runden
function deg(x)                       // Umwandlung von Bogenmaß in Gradmaß
function rad(x)                       // Umwandlung von Gradmaß in Bogenmaß
function Max6(a,b,c,d,e,f)            // Maximum von 6 Zahlen
function Min6(a,b,c,d,e,f)            // Minimum von 6 Zahlen
function Amax10(a,b,c,d,e,f,g,h,i,j)  // Absolutes Maximum von 10 Zahlen

function TPoint(tx,ty)                // Konstruktor für ein Punkt-Objekt (2-dim.)
function TPoint3(tx,ty,tz)            // Konstruktor für ein Punkt-Objekt (3-dim.)
function w2p(P9,kw,kb)                // World-To-Picture-Transformation von Punkt P9
function p2w(P9,kw,kb)                // Picture-To-World-Transformation von Punkt P9
function Schraeg(P9)                  // Schrägrisstransformation von Punkt P9
function initCanvas(kb,n,visa)        // Canvas initialisieren mit optionalem n-dim.KS
function clearImg(n,visa)             // Grafik löschen
function drawImg(name)                // Eine JPG-Grafikdatei laden
function drawCoord2(kw,kb)            // Koordinatensystem zeichnen (2-dim.)
function drawCoord3(kw,kb)            // Koordinatensystem zeichnen (3-dim.)
function drawPoint(tex,P9)            // Punkt P9 zeichnen und mit tex beschriften in schwarz
function killPoint(tex,P9)            // Punkt P9 zeichnen und mit tex beschriften in weiß
function drawPosText(x,y,tex)         // Text positioniert ausgeben (in Weltkoordinaten)
function drawText(y,tex)              // Text positioniert ausgeben (in Bildkoordinaten)
function drawWidth(f)                 // Liniendicke f
function drawColor(f)                 // Linienfarbe f
function fillColor(f)                 // Füllfarbe f
function drawCircle(M9,r9)            // Kreis zeichnen
function fillCircle(M9,r9)            // Kreis füllen
function fillArc(M9,r9,w1,w2)         // Kreissektor füllen
function drawLine(R9,S9)              // Strecke durch die Punkte R9 und S9 zeichnen
function drawLine3(W7,W8,W9)          // gefülltes Dreieck [W7,W8,W9,W7] zeichnen
function drawLine4(W6,W7,W8,W9)       // gefülltes Vierieck [W6,W7,W8,W9,W6] zeichnen
function drawLine1(R9,S9)             // Gerade durch die Punkte R9 und S9 zeichnen
function funEll(x,a,b)                // Funktionswert von Ellipse [a,b] mit Argument x
function drawEll(M9,a,b,r9)           // Ellipse [a,b] um M9(x/y) mit r9 = +/-1 zeichnen
function fillEll(M9,a,b,r9)           // Ellipse [a,b] um M9(x/y) mit r9 = +/-1 füllen

// Funktions-Plotter OHNE "parser.js" ------------------------------------------------
   var fpa, fpb, fpc, fpd;            // vier Funktionsparameter für 13 Funktionstypen
   var fselect = 1;                   // Funktions-Steuervariable (neu)
   var fselect_old = 0;               // Funktions-Steuervariable (alt)

   function fun01(x) { y = fpa*x + fpb; return y; }
   function fun02(x) { y = fpa*x*x + fpb*x + fpc; return y; }
   function fun03(x) { y = fpa*x*x*x + fpb*x*x + fpc*x + fpd; return y; }
   function fun04(x) { y = 1/(x - fpa); return y; }
   function fun05(x) { y = 1/((x - fpa)*(x - fpa)); return y; }
   function fun06(x) { y = fpb * Math.sqrt(fpa*x); return y; }
   function fun07(x) { y = (fpb/fpa) * Math.sqrt(fpa*fpa - x*x); return y; }
   function fun08(x) { y = (fpb/fpa) * Math.sqrt(x*x - fpa*fpa); return y; }
   function fun09(x) { y = fpa * Math.exp(fpb*x); return y; }
   function fun10(x) { y = fpa * Math.log(fpb*x); return y; }
   function fun11(x) { y = fpa * Math.sin(fpb*x + fpc); return y; }
   function fun12(x) { y = fpa * Math.exp(fpb*x) * Math.sin(fpc*x); return y; }
   function fun13(x) { y = fpa * Math.exp(-x*x/2) / Math.sqrt(2*Math.PI); return y; }

   function selectParam()             // Erzeugung der Funktions-Parameter
   function drawFunk(fselect)         // ausgewählte Funktionen zeichnen

// Funktions-Plotter MIT "parser.js" -------------------------------------------------
   function drawFun(kw,kb)
    // kw = Halbbreite des Koordinatensystems, kb = Canvas-Bildbreite
    // Funktion fun(s,x) zeichnen, benötigt "parser.js"
    // s = Funktionsterm, x = Argument

   function drawFun1(kw,kb,xa,xb,s)
    // kw = Halbbreite des Koordinatensystems, kb = Canvas-Bildbreite
    // Funktion fun(s,x) zeichnen, zwischen xa und xb, benötigt "parser.js"
    // s = Funktionsterm, x = Argument, benötigt "parser.js"

   function fillFun1(kw,kb,xa,xb,s)
    // kw = Halbbreite des Koordinatensystems, kb = Canvas-Bildbreite
    // Funktion fun(s,x) mit Farbe füllen, zwischen xa und xb, benötigt "parser.js"
    // s = Funktionsterm, x = Argument
```

Programmieren mit JavaScript, Teil 4

Ein universeller Formel-Parser (mit Anwendungen aus der Differenzial- und Integralrechnung)

[4.1] Der Formel-Parser

Der mathematische Formel-Parser „parser.js“ ist ein JavaScript-Programm, das von Raphael Graf entwickelt und von Herbert Paukert modifiziert wurde. Der Parser wertet einen String als eine mathematische Formel aus und verwendet dazu alle herkömmlichen mathematischen Operatoren für reelle Zahlen. Er kann auf unterschiedliche Art und Weise in eigene Programme eingebunden werden. Im vorliegenden Fall enthält er in seinem Code am Anfang zwei Funktionen, welche dann von außen aufgerufen werden können.

```
// "parser.js" Based on "ndef.parser" by Raphael Graf,
//  modified by Herbert Paukert, Version 4.0 am 10.12.2018.

var a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z;

function fun(s,x)  {
// Ein geparster Funktionterm "s" in nur EINER Variablen x.
   var t,z;
   t = Parser.parse(s);
   z = t.evaluate({ x: x });
   return z;
}

function formel(ss) {
// Eine geparste Formel "ss" in 26 Variablen a,b,c...,x,y,z.
   var tt,zz;
   tt = Parser.parse(ss);
   zz = tt.evaluate({ a:a,b:b,c:c,d:d,e:e,f:f,g:g,h:h,i:i,j:j,k:k,l:l,m:m,n:n,o:o,
                      p:p,q:q,r:r,s:s,t:t,u:u,v:v,w:w,x:x.y:y,z:z });
   return zz;
}

var Parser = (function (scope) {
       function object(o) {
              function F() {}
              F.prototype = o;
              return new F();
       }
       . . . . . .
       . . . . . .
       . . . . . .
       scope.Parser = Parser;
       return Parser;
})(typeof exports === 'undefined' ? {} : exports);

// End of "parser.js"
```

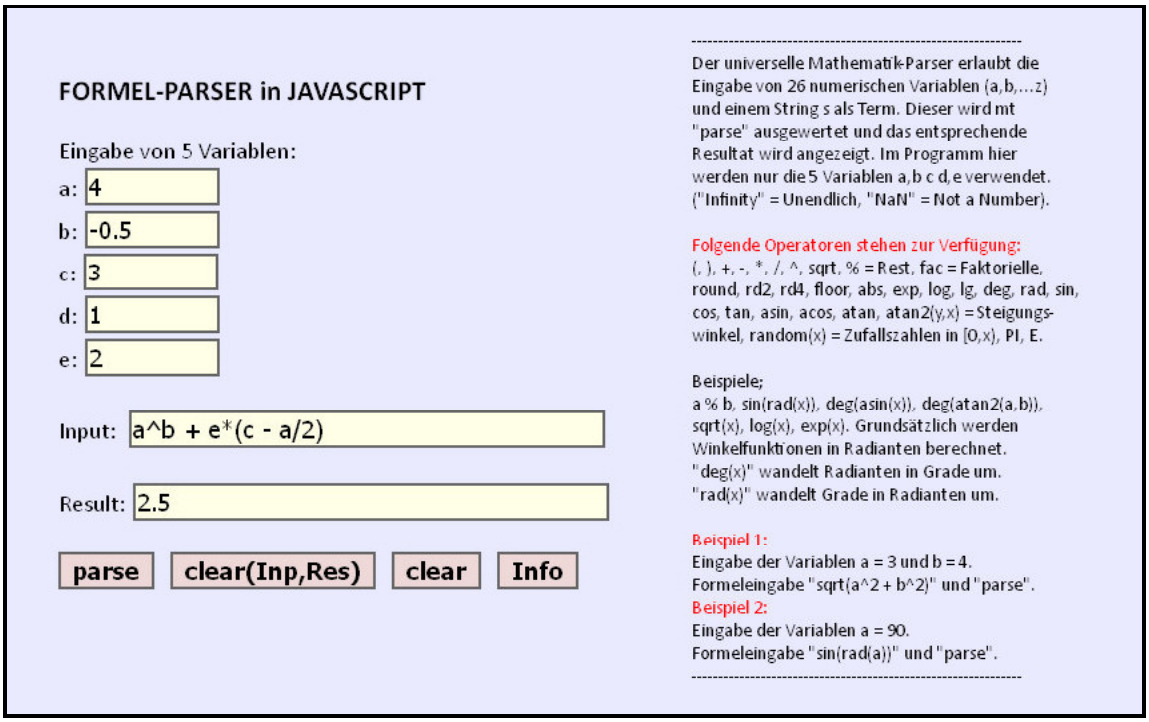
Die Funktion „***fun(s,x)“*** dient der Programmierung von einfachen mathematischen Funktionen, wobei der erste Parameter der Funktionsterm ist und der zweite Parameter das Funktionsargument. Die Funktion ***„formel(ss)“*** ermöglicht die Eingabe einer mathematischen Formel, welche die 26 Variablen ***a, b, c, . . ., x, y, z*** enthalten kann. Zurückgeliefert wird entweder eine relle Zahl oder bei fehlerhafter Eingabe „***NaN***" (Not a Number) bzw. "***Infinity***" (Unendlich).

Zulässige Operatoren sind: (,), +, -, *, /, ^, sqrt, % (=Rest), fac (=Faktorielle), round, rd2, rd4, floor, abs, exp, log, lg, deg, rad, sin, cos, tan, asin, acos, atan, atan2(y,x) (=Steigungswinkel), random(x) (=Zufallszahlen in [0,x)), PI (=Ludolfsche Zahl) und E (=Eulersche Zahl).

Beispiele: a % b, sin(rad(x)), deg(asin(x)), deg(atan2(a,b)), sqrt(x), log(x), exp(x), ……. Grundsätzlich werden alle Winkelfunktionen im Radianten-Maß berechnet. „deg(x)“ wandelt Radianten in Grade um. Hingegen wandelt „rad(x)“ Grade in Radianten um.

Der beschriebene Parser wird in allen folgenden Programmen verwendet. Diese stammen aus verschieden Gebieten der Differenzial- und Integralrechnung. Neben „***parser.js***“ gibt es noch „***mathe.js***“ und „***graph.js***“ als externe JavaScript-Dateien. „*mathe.js*“ enthält einschlägige mathematische Funktionen - „*graph.js*“ enthält einschlägige Grafikroutinen für das Canvas-Objekt.

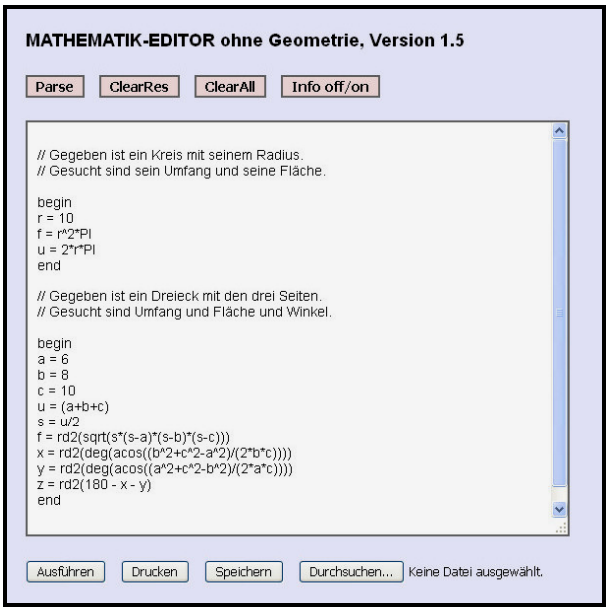
[4.1.1] Das Programm *„mparse.html“*

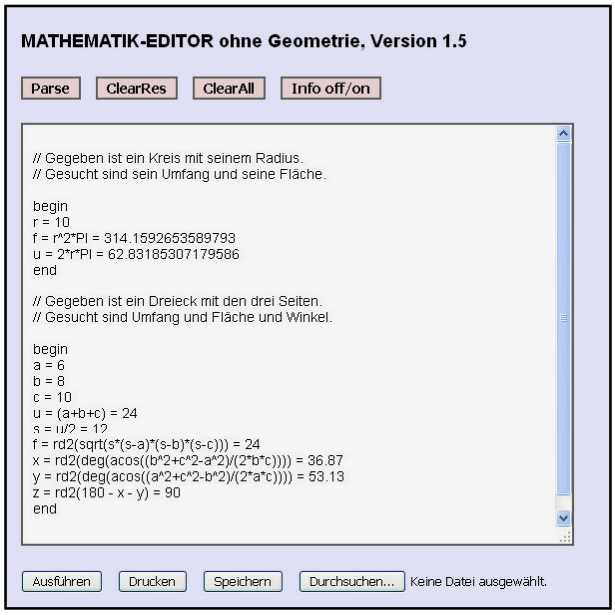



```
<!DOCTYPE html>
<html>
<head>
<title> mparse.html</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Formelparser">

<style type = "text/css">
  body {background-color:#EAEAFF; color:black; font-family:Calibri,Arial;
      font-size:18px; font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
  .cd1 {border: 2px solid #666666; background-color: #FFFFE8; font-size: 18px; margin:2px;}
  .cd2 {border: 2px solid #666666; background-color: #EED8D8; font-size: 18px; font-weight: bold;}
  #cd3 {visibility: hidden; font-size: 14px; }
</style>

<script src="parser.js"></script>

<script>
  function myTrim(x)  { return x.replace(/^\s+|\s+$/gm,'') }
  function round2(x)  { return Math.round(100*x)/100 }

  var hide = true;

  function Calcul(){
  // Formel berechnen
    document.MyForm.aus.value = '';
    a = 1*document.MyForm.ein1.value;
    b = 1*document.MyForm.ein2.value;
    c = 1*document.MyForm.ein3.value;
    d = 1*document.MyForm.ein4.value;
    e = 1*document.MyForm.ein5.value;

    s = document.MyForm.ein.value;
    s = myTrim(s);
    y = formel(s);   // => “parser.js”

    document.MyForm.aus.value = y;
  }

  function Clear(){
  // “Input” und “Result” löschen
    document.MyForm.ein.value = '';
    document.MyForm.aus.value = '';
  }
```

```
  function ClearAll(){
  // Alle Eingaben löschen
    document.MyForm.ein.value = '';
    document.MyForm.aus.value = '';
    document.MyForm.ein1.value = '';
    document.MyForm.ein2.value = '';
    document.MyForm.ein3.value = '';
    document.MyForm.ein4.value = '';
    document.MyForm.ein5.value = '';
    document.MyForm.edit.value = '';
  }

  function Info(){
  // Hilfstext ein/aus
    hide = !hide;
    if (hide == true) {
      document.getElementById("cd3").style.visibility = 'hidden';
    }
    else {
      document.getElementById("cd3").style.visibility = 'visible';
    }
  }
</script>
</head>

<body>
<form name="MyForm">
<h3>FORMEL-PARSER in JAVASCRIPT</h3>
Eingabe von 5 Variablen:<br>
a: <input type="text" name="ein1" value="" size="8" class="cd1"><br>
b: <input type="text" name="ein2" value="" size="8" class="cd1"><br>
c: <input type="text" name="ein3" value="" size="8" class="cd1"><br>
d: <input type="text" name="ein4" value="" size="8" class="cd1"><br>
e: <input type="text" name="ein5" value="" size="8" class="cd1"><br>
<br>
Input:  <input type="text" name="ein" value="" size="40" class="cd1"><br>
<br>
Result: <input type="text" name="aus" value=" " size="40" class="cd1" readonly><br>
<br>
<input type="button" name="btn" value="parse" class="cd2" onclick="Calcul()">  
<input type="button" name="btn" value="clear(Inp,Res)" class="cd2" onclick="Clear()">  
<input type="button" name="btn" value="clear" class="cd2" onclick="ClearAll()">  
<input type="button" name="btn" value="Info" class="cd2" onclick="Info()"><br>
<br>
<div id='cd3'>
<font color = "black">
-------------------------------------------------------------<br>
Der universelle Mathematik-Parser erlaubt die<br>
Eingabe von 26 numerischen Variablen (a,b,...z)<br>
und einem String s als Term. Dieser wird mt<br>
"parse" ausgewertet und das entsprechende<br>
Resultat wird angezeigt. Im Programm hier<br>
werden nur die 5 Variablen a,b c d,e verwendet.<br>
("Infinity" = Unendlich, "NaN" = Not a Number).<br>
<br>
<font color = "red">
Folgende Operatoren stehen zur Verfügung:<br>
</font>
(, ), +, -, *, /, ^, sqrt, % = Rest, fac = Faktorielle,<br>
round, rd2, rd4, floor, abs, exp, log, lg, deg, rad, sin,<br>
cos, tan, asin, acos, atan, atan2(y,x) = Steigungs-<br>
winkel, random(x) = Zufallszahlen in [0,x), PI, E.<br>
<br>
Beispiele;
<br>
a % b, sin(rad(x)), deg(asin(x)), deg(atan2(a,b)),<br>
sqrt(x), log(x), exp(x). Grundsätzlich werden<br>
Winkelfunktionen in Radianten berechnet.<br>
"deg(x)" wandelt Radianten in Grade um.<br>
"rad(x)" wandelt Grade in Radianten um.<br>
<br>
<font color = "red">Beispiel 1:</font><br>
Eingabe der Variablen a = 3 und b = 4. <br>
Formeleingabe "sqrt(a^2 + b^2)" und "parse".<br>
<font color = "red">Beispiel 2:</font><br>
Eingabe der Variablen a = 90. <br>
Formeleingabe "sin(rad(a))" und "parse".<br>
-------------------------------------------------------------<br>
</div>
</form>
</body>
</html>
```

[4.1.2] Das Programm *„matedit1.html“*





Das hier vorliegende Programm ist ein **Mathematik-Interpreter** mit dessen Hilfe einfache Mathematik-Skripts geschrieben, ausgeführt und auch als Textdateien gespeichert werden.
Der Mathematik-Editor erlaubt die Eingabe von 26 kleingeschriebenen Variablen (a,b,c, ...,z) und von mathematischen Ausdrücken in den Editorzeilen. Wird der Cursor auf diese platziert, dann werden die Textzeilen mit [**parse**] oder [**F1**] ausgewertet und die entsprechenden Rechenergebnisse werden rechts vom Gleichheitszeichen (=) angezeigt.
Hinweis 1: "Infinity" = Unendlich, "NaN" = Not a Number.
Hinweis 2: Kommentarzeilen (beginnend mit "//") werden nicht ausgewertet.
In den Kommenentaren nicht erlaubt sind alleinstehende Variablenbezeichner und Beistriche.
Die Kommentare dürfen nicht zwischen "begin" und "end" stehen.

Folgende Operatoren stehen zur Verfügung:

(,) , + , - , * , / , ^, sqrt, % = Rest, fac = Faktorielle, round, rd2, rd4 (= Runden), floor, abs, exp, log, lg, deg, rad, sin, cos, tan, asin, acos, atan, atan2(y,x) = Steigungswinkel, random(x) = Zufallszahlen in [0,x), PI (Ludolfsche Zahl) und E (Eulersche Zahl).

Beispiele: a % b, sin(rad(x)), deg(asin(x)), deg(atan2(a,b)), sqrt(x), log(x), exp(x).
Grundsätzlich werden alle Winkelfunktionen im Radianten-Maß berechnet.
"deg(x)" wandelt Radianten in Grade um. "rad(x)" wandelt Grade in Radianten um.

Folgende Script-Befehle stehen zur Verfügung:

Variable = Zahl, Variable = Ausdruck, Ausdruck =
Alle Script-Befehle, welche zwischen "**begin**" und "**end**" stehen, werden automatisch ausgeführt, wenn der Cursor auf "**begin**" platziert und dann [**Ausführen**] gedrückt wird.

Folgende Schalter stehen zur Verfügung:

[Parse] oder [F1] ... Ausführung einzelner Script-Befehle
[ClearRes] ... Löschung von allen Ergebniswerten
[ClearAll] ... Löschung von Text und Variablen
[Info off/on] …Aus- und Einblenden des Hilfstextes

[Ausführen] ... Script von "begin" bis "end" durchlaufen
[Drucken] ... Script ausdrucken
[Speichern] ... Speicherung in eine Textdatei
[Durchsuchen] ... Suchen und Laden einer Textdatei

```
<!DOCTYPE html>
<html>
<head>
<title>Mathematik-Editor (1)</title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Mathematik-Editor (1)">

<style>
body { background-color:#EAEAFF; color:black; font-family:Arial;
       font-size:18px; font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
.cd1 { border: 1px solid #666666; background-color: #FFFFE8; font-size: 14px;}
.cd2 { border: 2px solid #666666; background-color: #EED8D8; font-size: 14px; font-weight: bold;}
.cd3 { border: 2px solid #666666; padding: 10px; font-family:Arial; font-size:14px;}
#cd4 { position:absolute; top: 20px; left: 700px; font-family: Arial; font-size: 12px;}
</style>

<script src="parser.js"></script>
<script src="mathe.js"></script>

<script>

  /* 26 numeriche Variable a,b,c ..., x,y,z sind in "parser.js" definiert */

  var help = true;

  function setValues(vari,value) {
    if (vari == 'a') { a = value; }
    if (vari == 'b') { b = value; }
    if (vari == 'c') { c = value; }
    if (vari == 'd') { d = value; }
    if (vari == 'e') { e = value; }
    if (vari == 'f') { f = value; }
    if (vari == 'g') { g = value; }
    if (vari == 'h') { h = value; }
    if (vari == 'i') { i = value; }
    if (vari == 'j') { j = value; }
    if (vari == 'k') { k = value; }
    if (vari == 'l') { l = value; }
    if (vari == 'm') { m = value; }
    if (vari == 'n') { n = value; }
    if (vari == 'o') { o = value; }
    if (vari == 'p') { p = value; }
    if (vari == 'q') { q = value; }
    if (vari == 'r') { r = value; }
    if (vari == 's') { s = value; }
    if (vari == 't') { t = value; }
    if (vari == 'u') { u = value; }
    if (vari == 'v') { v = value; }
    if (vari == 'w') { w = value; }
    if (vari == 'x') { x = value; }
    if (vari == 'y') { y = value; }
    if (vari == 'z') { z = value; }
  }

  function clearVar() {
    a = 0; b = 0; c = 0; d = 0; e = 0; f = 0; g = 0; h = 0; i = 0; j = 0; k = 0; l = 0; m = 0;
    n = 0; o = 0; p = 0; q = 0; r = 0; s = 0; t = 0; u = 0; v = 0; w = 0; x = 0; y = 0; z = 0;
  }

  function myTrim(x) {return x.replace(/^\s+|\s+$/gm,'')}  // Entfernt alle randständigen Blanks

  function loadText() {
  /* Textdatei laden */
    var txtbox = document.getElementById('edit');
    var fileToLoad = document.getElementById('fileInput').files[0];
    var reader = new FileReader();
    textType = /text.*/;
    if ( !fileToLoad.type.match(textType) ) {
      document.getElementById('fileInput').value = '';
      alert('Falscher Dateityp');
      return;
    }
    reader.readAsText(fileToLoad);
    reader.addEventListener('load', function() {
      txtbox.value = reader.result;
    },false);
  }
```

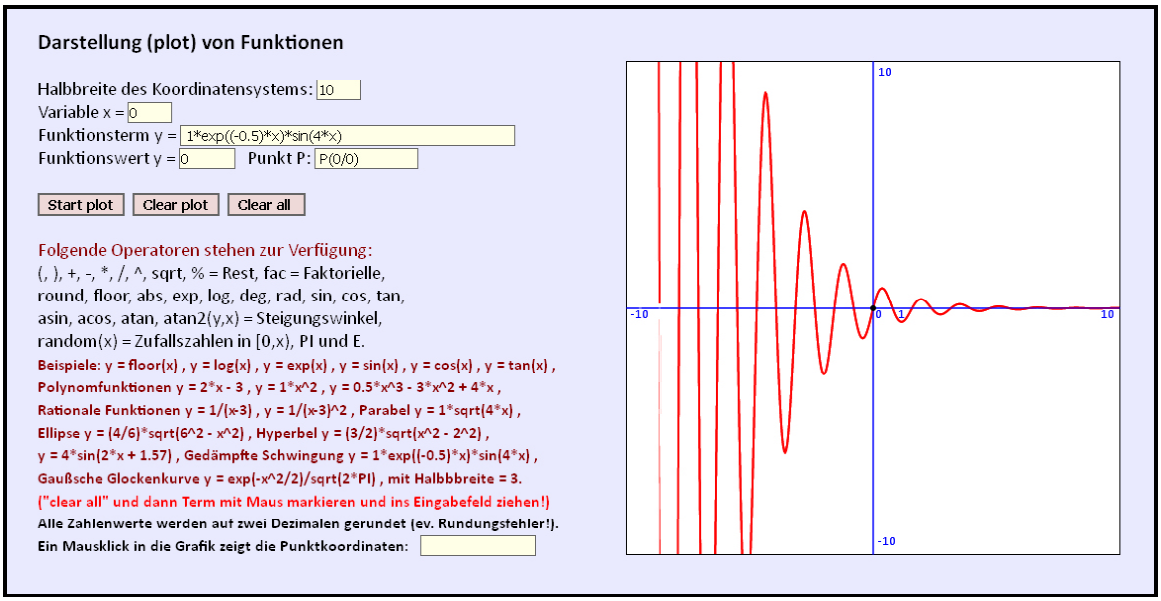
```
function saveText() {
/* Textdatei speichern */
  document.getElementById('fileInput').value = '';
  ext = '.txt';
  fname = 'text';
  textFile = document.getElementById('edit').value;
  var textBlob = new Blob([textFile], {type:'text/plain'});
  var link = document.createElement('a');
  link.href = window.URL.createObjectURL(textBlob);
  fname = prompt('Dateiname ohne Extension',fname);
  fname = fname + ext;
  link.download = fname;
  document.body.appendChild(link);
  link.click();
  document.body.removeChild(link);
}

function KeyListener(ev) {
/* Tastatur-Eventhandler */
  var taste = ev.keyCode || ev.which;
  if (taste == 112) { Calcul(); }                        // Funktionstaste F1
}

function Calcul() {
/* Script-Befehle im Einzelschritt-Modus */
/* Ermittelt die aktuelle Cursorposition (rowPos,colPos) im Text. */
/* Ermittelt auch die aktuelle Textzeile (rowText) */
/* und erzeugt links und rechts vom "=" stehende Texte (lText,rText). */
/* Dann werden die Texte geparst, d.h. Variable und Ausdrücke ausgewertet. */
/* Zuletzt wird der gesamte Text mit den Auswertungen neu erzeugt. */

  const LF = '\n';                                       // LineFeed
  const GL = '=';                                        // Gleichheit
  const KT = '//';                                       // Kommentar

  var el  = document.getElementById("edit");             // Textarea
  var tex = el.value;                                    // Gesamter Text
  var len = el.value.length;                             // Gesamte Textlänge
  var tar0 = tex.split(LF);                              // Text -> Array
  var countLF = tar0.length;                             // Gesamte Zeilenanzahl

  var startPos  = el.selectionStart;                     // Cursorposition im Text (Start)
  var texTo   = tex.substring(0,startPos);               // Text von 0 bis Start
  var tar1    = texTo.split(LF);                         // Text -> Array
  var rowPos  = tar1.length;                             // aktuelle Zeilenposition !!!
  rowText = myTrim(tar0[rowPos-1]);                      // aktueller Zeilentext !!!
  var lastLF  = texTo.lastIndexOf(LF);                   // Letzter Linefeed im Text
  var colPos  = startPos - lastLF;                       // aktuelle Spaltenposition !!!
  var posi = rowText.indexOf(KT);                        // Kommentarzeile abfragen
  if (posi >= 0) { return; }

  rowText.toLowerCase();
  posi = rowText.indexOf(GL);
  if (posi > 0) {
    lText = myTrim(rowText.substring(0,posi));           // linken und rechten Text
    rText = myTrim(rowText.substring(posi+1));           // der Textzeile ermitteln
    setValues(lText,rText);                              // Variableneingabe "var = "
  }
  else { return; }

  if ( lText.length != 1 ) { rText = ''; }               // linker Text = math. Term
  if ( lText != '') { yl = formel(lText); }              // linken Text parsen
  if ( rText != '' ) {                                   // rechten Text parsen
    yr = formel(rText);                                  // und auf die Variable
    if ( lText.length == 1 ) { setValues(lText,yr); }    // des linken Textes abspeichern
  }
  else { yr = yl; }
  if ( isNaN(rText) ) {                                  // Parser-Auswertungen
    tt = lText + ' = ' + rText + ' = ' + yr;
  }
  else { tt = lText + ' = ' + yr; }

  tar0[rowPos-1] = tt;                                   // gesamten Text neu ausgeben
  tex = tar0.toString();
  tex = tex.replace(/,/g,'\n');
  el.value = tex;
  el.focus();
}
```

```
function runScript() {
/* Script-Befehle im Automatik-Modus */
  const LF = '\n';
  const GL = '=';
  const KT = '//';
  var el  = document.getElementById("edit");
  var tex = el.value;
  var tar0 = tex.split(LF);
  var startPos  = el.selectionStart;
  var texTo   = tex.substring(0,startPos);
  var tar1    = texTo.split(LF);
  var rowPos  = tar1.length;
  rowText = myTrim(tar0[rowPos-1]);
  rowText.toLowerCase();
  var num = rowPos-1;
  if ( rowText != 'begin' ) { return; }
  do {                                                  // Befehlsschlange starten
    num = num + 1;
    rowText = myTrim(tar0[num]);
    var posi = rowText.indexOf(KT);
    if (posi >= 0) { return; }
    rowText.toLowerCase();
    posi = rowText.indexOf(GL);
    if (posi > 0) {
      lText = myTrim(rowText.substring(0,posi));
      rText = myTrim(rowText.substring(posi+1));
      setValues(lText,rText);
      if ( lText.length != 1 ) { rText = ''; }
      if ( lText != '') { yl = formel(lText); }
      if ( rText != '' ) {
        yr = formel(rText);
        if ( lText.length == 1 ) { setValues(lText,yr); }
      }
      else { yr = yl; }
      if ( isNaN(rText) ) {
        tt = lText + ' = ' + rText + ' = ' + yr;
      }
      else { tt = lText + ' = ' + yr; }
      tar0[num] = tt;
      tex = tar0.toString();
      tex = tex.replace(/,/g,'\n');
      el.value = tex;
    }
  } while (rowText != 'end');                           // Befehlsschlange beenden
  el.focus();
}
function clearResults() {
/* alle Ergebnis-Werte ab dem letzten "=" löschen */
/* ausgenommen sind die direkten Variablenzuweisungen */
  const LF = '\n';
  const GL = '=';
  var el   = document.getElementById("edit");
  var tex  = el.value;
  var len  = el.value.length;
  var tar0 = tex.split(LF);
  var anz  = tar0.length;
  for (var i = 0; i < anz; i++) {
    var ss = myTrim(tar0[i]);
    if ( ss != '' ) {
      var p1 = ss.indexOf(GL);
      var p2 = ss.lastIndexOf(GL);
      if ( (p1 > 0) && (p1 != p2) ) {
        lText = myTrim(ss.substring(0,p1));
        rText = myTrim(ss.substring(p1+1));
        if ( !isNaN(rText) ) {
          var tt = ss.substring(0,p1+1);
          tar0[i] = tt;
        }
        else {
          if ( p2 > p1 ) {
            var tt = ss.substring(0,p2);
            tar0[i] = tt;
          }
        }
      }
    }
  }
  tex = tar0.toString();
  tex = tex.replace(/,/g,'\n');
  el.value = tex;
  el.focus();
}
```

```
  function printText() {
  /* Text drucken */
    var text = document.MyForm.edit.value;
    w = window.open(","location=no,scrollbars=no,menubar=no");
    w.document.open();
    w.document.write("<html><body><pre>");
    w.document.write(text);
    w.document.write("</pre></html></body>");
    w.document.close();
  }

  function clearAll() {
  /* Text und Variable löschen */
    var el = document.getElementById("edit");
    el.value = '';
    clearVar();
    el.focus();
  }

  function info() {
  /* Hilfe ein/aus */
     help = !help;
     if (help) { document.getElementById('cd4').style.visibility = 'visible'; }
     if (!help) { document.getElementById('cd4').style.visibility = 'hidden'; }
  }

</script>
</head>

<body onkeydown = "KeyListener(event)">

<form name="MyForm">
<h4>MATHEMATIK-EDITOR ohne Geometrie, Version 1.5</h4>
<input type="button" name="btn1" value="Parse" class="cd2" onclick="Calcul()">  
<input type="button" name="btn2" value="ClearRes" class="cd2" onclick="clearResults()">  
<input type="button" name="btn3" value="ClearAll" class="cd2" onclick="clearAll()">  
<input type="button" name="btn4" value="Info off/on" class="cd2" onclick="info()">  
<br><br>

<textarea name="edit" id="edit" class="cd3" value="" cols="80" rows="24" wrap="off">

// Gegeben ist ein Kreis mit seinem Radius.
// Gesucht sind sein Umfang und seine Fläche.

begin
r = 10
f = r^2*PI
u = 2*r*PI
end

// Gegeben ist ein Dreieck mit den drei Seiten.
// Gesucht sind Umfang und Fläche und Winkel.

begin
a = 6
b = 8
c = 10
u = (a+b+c)
s = u/2
f = rd2(sqrt(s*(s-a)*(s-b)*(s-c)))
x = rd2(deg(acos((b^2+c^2-a^2)/(2*b*c))))
y = rd2(deg(acos((a^2+c^2-b^2)/(2*a*c))))
z = rd2(180 - x - y)
end

</textarea>

<br><br>
<input type="button" value="Ausführen" onclick="runScript()">  
<input type="button" value="Drucken" onclick="printText()">  
<input type="button" value="Speichern" onclick="saveText()">  
<input type="file" id="fileInput" onchange="loadText()">  
<br>
```

```
<div id='cd4'>
<font color = "black">

<font color = "darkred"><b><i>Der Mathematik-Editor </b></i></font>
(&copy Herbert Paukert) erlaubt die Eingabe von<br>
26 <u>kleingeschriebenen</u> Variablen (a, b, c, . . ., z) und von mathematischen<br>
Ausdrücken in den Editorzeilen. Wird der Cursor auf diese platziert,<br>
dann werden die Textzeilen mit <b>[parse]</b> oder <b>[F1]</b> ausgewertet und<br>
die entsprechenden Rechenergebnisse werden im Texteditor angezeigt.<br>
<br>

Hinweis 1: "Infinity" = Unendlich, "NaN" = Not a Number.<br>
Hinweis 2: Kommentarzeilen (beginnend mit "//") werden nicht ausgewertet.<br>
In Kommentarzeilen nicht erlaubt sind Variablenbezeichner und Beistriche.<br>
Kommentarzeilen dürfen nicht zwischen "begin" und "end" stehen.<br>
<br>

<font color = "darkred">
<b><i>
Folgende Operatoren stehen zur Verfügung:<br>
</b></i>
</font>
<br>
(, ), +, -, *, /, ^, sqrt, % = Rest, fac = Faktorielle, round,rd2,rd4 (Runden), floor, abs,<br>
exp, log, lg, deg, rad, sin, cos, tan, asin, acos, atan, atan2(y,x) = Steigungswinkel, <br>
random(x) = Zufallszahlen in [0,x), PI (Ludolfsche Zahl) und E (Eulersche Zahl).<br>
<br>

<b>Beispiele:</b> a % b, sin(rad(x)), deg(asin(x)), deg(atan2(a,b)), sqrt(x), log(x), exp(x)<br>
Grundsätzlich werden alle Winkelfunktionen im Radianten-Maß berechnet.<br>
"deg(x)" wandelt Radianten in Grade um. "rad(x)" wandelt Grade in Radianten um.<br>
<br>

<font color = "darkred">
<b><i>
Folgende Script-Befehle stehen zur Verfügung:<br>
</b></i>
</font>
<br>
<b>Variable = Zahl,   Variable = Ausdruck,   Ausdruck = </b><br>
Alle Script-Befehle zwischen <b>"begin"</b> und <b>"end"</b> werden automatisch ausgeführt,<br>
wenn der Cursor auf <b>"begin"</b> platziert und dann <b>[Ausführen]</b> gedrückt wird.<br>
<br>

<font color = "darkred">
<b><i>
Folgende Schalter stehen zur Verfügung:<br>
</b></i>
</font>
<br>
<b>[Parse] oder [F1] ... Ausführung der Script-Befehle</b><br>
[ClearRes] ... Löschung von allen Ergebniswerten<br>
[ClearAll] ... Löschung von Text und Variablen<br>
<br>
<b>[Ausführen] ... Script von "begin" bis "end" durchlaufen</b><br>
[Drucken] ... Script ausdrucken<br>
[Speichern] ... Speicherung in eine Textdatei<br>
[Durchsuchen] ... Suchen und Laden einer Textdatei<br>

</font>
</div>

</form>
</body>
</html>
```

[4.1.3] Das Programm *„fplot.html“*



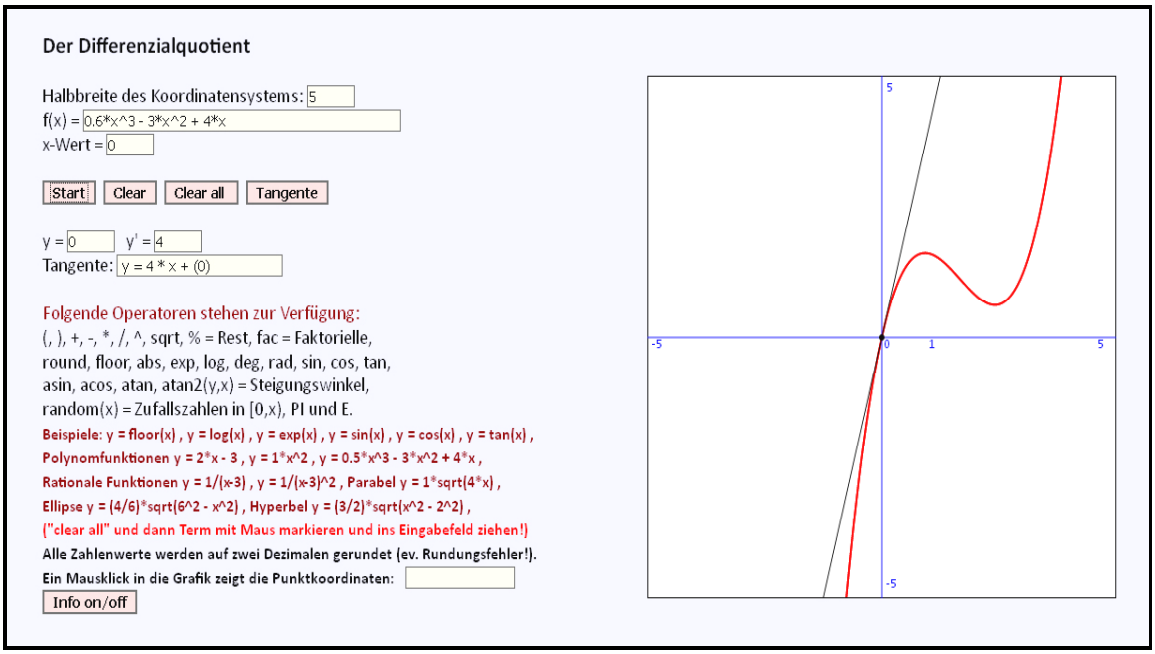

```
<!DOCTYPE html>
<html>
<head>
<title> </title>
<meta charset="ISO-8859-1">
<!-- <meta name="viewport" content="width=device-width, initial-scale=1.0"> -->
<meta name="author" content="Paukert Herbert">

<style type = "text/css">
body {background-color:#EAEAFF; color:black; font-family:Calibri,Arial; font-size:19px;
      font-weight:normal; text-align:left; padding-left:2%; padding-top:1%; }
.cd1 {border: 1px solid #666666; background-color: #FFFFE8; font-size: 15px;}
.cd2 {border: 2px solid #666666; background-color: #EED8D8; font-size: 16px; font-weight: bold;}
</style>

<script src="parser.js"></script>
<script src="graph.js"></script>

<script>
  O = new TPoint(0,0);                                    // ==> "graph.js"
  A = new TPoint(0,0);

function Clear1() {
  // Grafik löschen
  document.MyForm.aus0.value = ' ';
  document.MyForm.aus1.value = ' ';
  document.MyForm.aus4.value = ' ';
  initCanvas(kb,2,1);                                     // ==> "graph.js"
}

function Clear2() {
  // Grafik löschen
  document.MyForm.ein1.value = 0;
  document.MyForm.ein2.value = ' ';
  document.MyForm.aus0.value = ' ';
  document.MyForm.aus1.value = ' ';
  document.MyForm.aus4.value = ' ';
  initCanvas(kb,2,1);                                     // ==> "graph.js"
}

function Calcul() {
  // Hauptroutine
  kw = 1 * document.MyForm.ein0.value;
  document.MyForm.aus0.value = ' ';
  document.MyForm.aus1.value = ' ';
  document.MyForm.aus4.value = ' ';
```

```
  x = parseInt( document.MyForm.ein1.value);
  s = document.MyForm.ein2.value;
  s = myTrim(s);
  y = fun(s,x);                                          // ==> "parser.js"
  y = round2(y);
  document.MyForm.aus0.value = y;
  A.x = x;
  A.y = y;
  drawFun(kw,kb,s);                                      // ==> "graph.js"
  drawPoint("",A);                                       // ==> "graph.js"
  texA = ' P(' + A.x + '/' + A.y + ')';
  document.MyForm.aus1.value = texA;
  l = round2(A.len());                                   // ==> "graph.js"
  w = round2(A.win());
}
</script>
</head>

<body onload = "initCanvas(kb,2,1)">

<div id='dc' style="position:absolute; top:60px; left:650px ">
  <canvas id='MyCanvas' width='500' height='500' onmousedown = "showPos(event)">
     Your browser does not support HTML5 Canvas.
  </canvas>
</div>

<form name="MyForm">
<h3>Darstellung (plot) von Funktionen</h3>
Halbbreite des Koordinatensystems: <input type="text" name="ein0" value="10" size="3"
class="cd1"><br>
Variable x = <input type="text" name="ein1" value="1" size="3" class="cd1"><br>
Funktionsterm y = <input type="text" name="ein2" value="x^2" size="45" class="cd1"><br>
Funktionswert y = <input type="text" name="aus0" value=" " size="5" class="cd1"
readonly="true">  
Punkt P: <input type="text" name="aus1" value=" " size="13" class="cd1" readonly="true"><br>

<br>
<input type="button" name="calc1" value="Start plot" class="cd2" onclick="Calcul()"> 
<input type="button" name="calc2" value="Clear plot" class="cd2" onclick="Clear1()"> 
<input type="button" name="calc2" value="Clear all " class="cd2" onclick="Clear2()"><br>
<br>

<b><font size="-0.80">
<font color="darkred">
Folgende Operatoren stehen zur Verfügung:<br>
(, ), +, -, *, /, ^, sqrt, % = Rest, fac, round, floor, abs, exp, log, deg, rad, sin, cos, tan,<br>
asin, acos, atan, atan2(y,x) = Steigungswinkel, random(x) = Zahlen in [0,x), PI und E.<br>
</font>
<font color = "black">
Beispiele von Funktionen:<br>
y = abs(x) , y = floor(x) , y = log(x) , y = exp(x) , y = sin(x) , y = cos(x) , y = tan(x) ,<br>
Polynomfunktionen y = 2*x - 3 , y = 1*x^2 , y = 0.5*x^3 - 3*x^2 + 4*x ,<br>
Rationale Funktionen y = 1/(x - 3) , y = x^3/(x^2 - 4) , Parabel y = 1*sqrt(4*x) ,<br>
Ellipse y = (4/6)*sqrt(6^2 - x^2) , Hyperbel y = (3/2)*sqrt(x^2 - 2^2).<br>
Gaußsche Glockenkurve y = exp(-x^2/2)/sqrt(2*PI) , mit Halbbbreite = 3.<br>
<font color = "darkred">
Allgemeine Hinweise:<br>
"clear all" und dann Term mit Maus markieren und ins Eingabefeld ziehen!<br>
Alle Zahlenwerte werden auf zwei Dezimalen gerundet (ev. Rundungsfehler).<br>
Ein Mausklick in die Grafik zeigt die Punktkoordinaten:  
<input type="text" name="aus4" value=" " size="16" class="cd1" readonly><br>
</font>
</b></font>

<b><font size="-1" color="darkblue">&copy Herbert Paukert</font></b>
</form>
<script>
   function showPos(ev) {
     kw = 1 * document.MyForm.ein0.value;
     x1 = ev.clientX - 650;
     y1 = ev.clientY - 60;
     x0 = x1*2*kw/kb - kw;
     y0 = (kb - y1)*2*kw/kb - kw;
     info = round2(x0) + '  /  ' + round2(y0);
     document.MyForm.aus4.value = info;
   }
</script>
</body>
</html>
```

[4.2] Differenzialrechnung

[4.2.1] Das Programm *„diffquot.html“*

```
-------------------------------------------------------------------------
Gegeben ist eine Funktion y = f(x) auf dem Intervall [a,b].
Der Differenzialquotient (Ableitung) dy/dx = y' = f'(x) an der
Stelle x ist der Anstieg (k) der Kurventangente an dieser Stelle.

Geradengleichung der Tangente im Punkt P(a/f(a)): y = k * x + d.

Dabei gilt: k = f'(a), f(a) = f'(a)*a + d, d = f(a) - f'(a)* a.
Einsetzen in die Geradengleichung ergibt: y = f'(a)*(x - a) + f(a).
y = f'(a)*(x - a) + f(a) ist Tangentengleichung im Punkt P(a/f(a)).
-------------------------------------------------------------------------
```



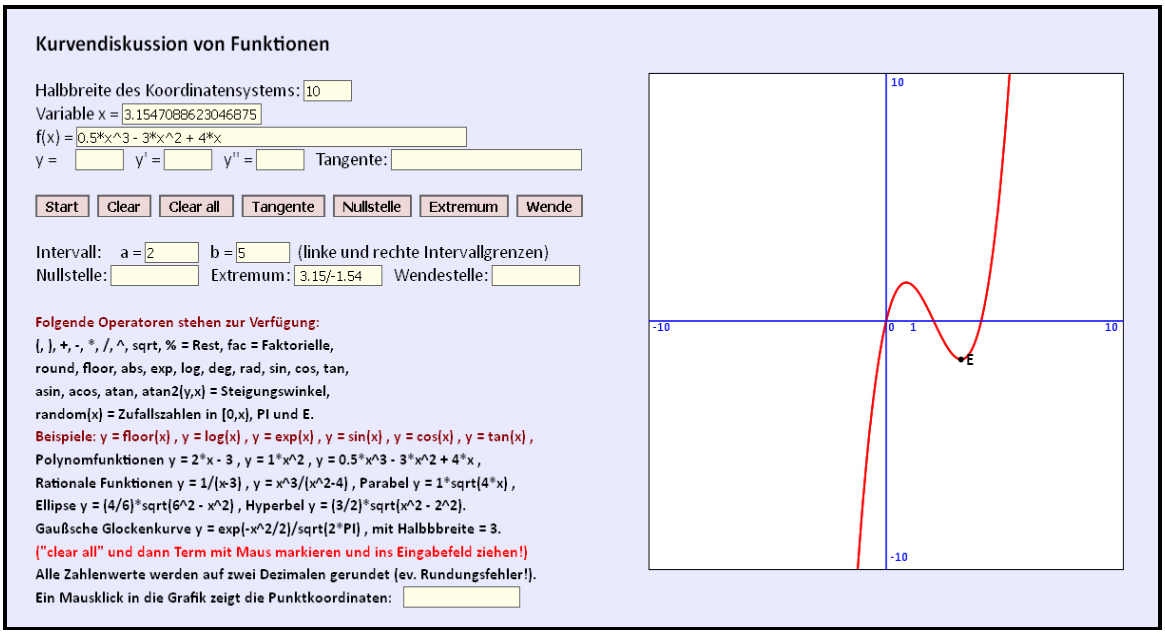

```
<!DOCTYPE html>
<html>

<head>
<title> Differenzialqutient </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" Differenzialqutient ">

<style type = "text/css">
body {background-color:#EAEAFF; color:black; font-family:Calibri,Arial;
      font-size:19px; font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
.cd1 {border: 1px solid #666666; background-color: #FFFFE8; font-size: 14px;}
.cd2 {border: 2px solid #666666; background-color: #EED8D8; font-size: 14px; font-weight: bold;}
</style>

<script src="parser.js"></script>
<script src="graph.js"></script>
<script src="mathe.js"></script>
```

```
<script>

  var hide = true;
  var clear = true;
  var tng = false;
  var run = 0;
  var s;  // Funktionsterm
  A = new TPoint(0,0);
  B = new TPoint(0,0);
  C = new TPoint(0,0);
  D = new TPoint(0,0);
  E = new TPoint(0,0);
  O = new TPoint(0,0);
  T = new TPoint(0,0);

  function Clear1() {
  // Grafik löschen
    document.MyForm.aus0.value = ' ';
    document.MyForm.aus4.value = ' ';
    document.MyForm.aus5.value = ' ';
    document.MyForm.aus6.value = ' ';
    clear = true;
    initCanvas(kb,2,1);
    clear = false;
    document.MyForm.calc1.focus();
  }

  function Clear2() {
  // Grafik löschen
    document.MyForm.ein1.value = 0;
    document.MyForm.ein2.value = ' ';
    document.MyForm.aus0.value = ' ';
    document.MyForm.aus4.value = ' ';
    document.MyForm.aus5.value = ' ';
    document.MyForm.aus6.value = ' ';
    clear = true;
    initCanvas(kb,2,1);
    clear = false;
    document.MyForm.calc1.focus();
  }

function Calcul() {
// Hauptroutine
  kw = 1 * document.MyForm.ein0.value;
  initCanvas(kb);
  document.MyForm.aus0.value = ' ';
  document.MyForm.aus4.value = ' ';
  document.MyForm.aus5.value = ' ';
  document.MyForm.aus6.value = ' ';

  x = 1 * document.MyForm.ein1.value;
  s = document.MyForm.ein2.value;
  s = myTrim(s);
  y = fun(s,x);
  document.MyForm.aus0.value = y;
  A.x = x;
  A.y = y;
  drawFun(kw,kb,s);                                    // ==> "graph.js"
  drawPoint('',A);

  O.x = 0;
  O.y = 0;
  C.x = 0;
  C.y = 0;
  D.x = 0;
  D.y = 0;

  l = A.len();
  l = round2(l);
  w = A.win();
  w = round2(deg(w));

  x = 1 * document.MyForm.ein1.value;
  s = document.MyForm.ein2.value;
  ky = differ(s,x);                                    // ==> "mathe.js"
  ky = round2(ky);
  document.MyForm.aus5.value = round2(ky);
  // alert('Betrag OA = ' + l + ', Winkel OA = ' + w);
```

```
if (tng)  {
  drawWidth(1);
  drawColor("#000000");
  T = tangente(s,A);                                    // ==> "mathe.js"
  texT = ' y = ' + T.x + ' * x + (' + T.y + ')';
  if (Number.isNaN(T.y)) { texT = ' x = ' + T.x; }
  if (T.y == Infinity) { texT = ' x = ' + T.x; }

  document.MyForm.aus6.value = texT;

  if ( Number.isNaN(A.y) ) {
    A.y = 0;
    drawPoint("",A);
    document.MyForm.aus6.value = '';
  }
  else {
  if ( (Number.isNaN(T.y)) || (T.y == Infinity) ) {
    B.x = A.x;
    B.y = -0.999;
    //  drawPoint("B",B);
    drawLine1(A,B);
  }
  else {
    B.x = A.x - 0.999;
    B.y = T.x * B.x + T.y;
    //  drawPoint("B",B);
    drawLine1(A,B);
  }
  }
  }
  }

  function Tang() {
    tng = true;
    Calcul();
    tng = false;
  }

  function Info(){
    hide = !hide;
    if (hide) {
      document.getElementById("dc").style.visibility = 'visible';
      document.getElementById("cd3").style.visibility = 'hidden';
    }
    if (!hide) {
      document.getElementById("dc").style.visibility = 'hidden';
      document.getElementById("cd3").style.visibility = 'visible';
    }
  }

</script>
</head>

<body onload = "initCanvas(kb,2,1);">
<div id='dc' style="position:absolute; top:70px; left:700px ">
  <canvas id='MyCanvas' width='500' height='500' onmousedown = "showPos(event)">
     Your browser does not support HTML5 Canvas.
  </canvas>
</div>

<form name="MyForm">
<h3>Der Differenzialquotient</h3>
Halbbreite des Koordinatensystems: <input type="text" name="ein0" value="5" size="4" class="cd1"><br>
f(x) = <input type="text" name="ein2" value="0.6*x^3 - 3*x^2 + 4*x" size="52" class="cd1"><br>
x-Wert = <input type="text" name="ein1" value="0" size="4" class="cd1"><br>
<br>
<input type="button" name="calc1" value="Start" class="cd2" onclick="Calcul()"> 
<input type="button" name="calc2" value="Clear" class="cd2" onclick="Clear1()"> 
<input type="button" name="calc3" value="Clear all " class="cd2" onclick="Clear2()"> 
<input type="button" name="tang" value="Tangente" class="cd2" onclick="Tang()"><br>
<br>
y = <input type="text" name="aus0" value=" " size="4" class="cd1" readonly = "true">
  
y' = <input type="text" name="aus5" value=" " size="4" class="cd1" readonly = "true"><br>
Tangente: <input type="text" name="aus6" value=" " size="25" class="cd1" readonly = "true">
<br>
<br>
```

```
<b>
<font color="black" size="-0.80">
Folgende Operatoren stehen zur Verfügung:<br>
 (, ), +, -, *, /, ^, sqrt, % = Rest, fac = Faktorielle, <br>
round, rd2, rd4, floor, abs, exp, log, lg, deg, rad, sin, cos, tan, <br>
asin, acos, atan, atan2(y,x) = Steigungswinkel,<br>
random(x) = Zufallszahlen in [0,x), PI und E.<br>
<font color = "darkred">
Beispiele: y = floor(x) , y = log(x) , y = exp(x) , y = sin(x) , y = cos(x) , y = tan(x) ,<br>
Polynomfunktionen y = 2*x - 3 , y = 1*x^2 , y = 0.5*x^3 - 3*x^2 + 4*x ,<br>
Rationale Funktionen y = 1/(x-3) , y = 1/(x-3)^2 , Parabel y = 1*sqrt(4*x) ,<br>
Ellipse y = (4/6)*sqrt(6^2 - x^2) , Hyperbel y = (3/2)*sqrt(x^2 - 2^2) ,<br>
</font>
<font color = "red">
("clear all" und dann Term mit Maus markieren und ins Eingabefeld ziehen!)<br>
</font>
<font color = "black">
Alle Zahlenwerte werden auf zwei Dezimalen gerundet (ev. Rundungsfehler!).<br>
Ein Mausklick in die Grafik zeigt die Punktkoordinaten:  
<input type="text" name="aus4" value=" " size="15" class="cd1" readonly><br>
<input type="button" name="btn" value="Info on/off" class="cd2" onclick="Info()"><br>
</font>
</font>
</b>

</form>

<div id='cd3' style="position:absolute; top:70px; left:700px; visibility:hidden;">
<font color = "black">
<br><br>
-------------------------------------------------------------------------------------<br>
Gegeben ist eine Funktion y = f(x) auf dem Intervall [a,b].<br>
Der Differenzialquotient (Ableitung) dy/dx = y' = f'(x) an der<br>
Stelle x ist der Anstieg (k) der Kurventangente an dieser Stelle.<br>
<br>
Geradengleichung der Tangente im Punkt P(a/f(a)): y = k * x + d.<br>
<br>
Dabei gilt: k = f'(a), f(a) = f'(a)*a + d, d = f(a) - f'(a)* a.<br>
Einsetzen in die Geradengleichung ergibt: y = f'(a)*(x - a) + f(a).<br>
<b>y = f'(a)*(x - a) + f(a)</b> ist Tangentengleichung im Punkt P(a/f(a)).<br>
-------------------------------------------------------------------------------------<br>
<br>
</font>
</div>

<script>
   function showPos(ev) {
   kw = 1 * document.MyForm.ein0.value;
   x1 = ev.clientX - 700;
   y1 = ev.clientY - 70;
   x0 = x1*2*kw/kb - kw;
   y0 = (kb - y1)*2*kw/kb - kw;
   info = round2(x0) + '  /  ' + round2(y0);
   document.MyForm.aus4.value = info;
   }
   document.MyForm.calc1.focus();
</script>
</body>
</html>
```

[4.2.2] Das Programm *„kudi.html“*




```
<!DOCTYPE html>
<html>
<head>
<title> Kurvendiskussion </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" Kurvendiskussion ">

<style type = "text/css">
body {background-color:#EAEAFF; color:black; font-family:Calibri,Arial;
      font-size:19px; font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
.cd1 {border: 1px solid #666666; background-color: #FFFFE8; font-size: 14px;}
.cd2 {border: 2px solid #666666; background-color: #EED8D8; font-size: 14px; font-weight: bold;}
</style>

<script src="parser.js"></script>
<script src="graph.js"></script>
<script src="mathe.js"></script>

<script>

  var tng = false;
  var run = 0;
  var s; // Funktionsterm

  A = new TPoint(0,0);
  B = new TPoint(0,0);
  C = new TPoint(0,0);
  D = new TPoint(0,0);
  E = new TPoint(0,0);
  W = new TPoint(0,0);
  O = new TPoint(0,0);
  T = new TPoint(0,0);

  function Clear1() {
  // Grafik löschen
    document.MyForm.ein1.value = 0;
    document.MyForm.aus0.value = ' ';
    document.MyForm.aus4.value = ' ';
    document.MyForm.aus5.value = ' ';
    document.MyForm.aus5a.value = ' ';
    document.MyForm.aus6.value = ' ';
    document.MyForm.aus6a.value = ' ';
    document.MyForm.aus6b.value = ' ';
    document.MyForm.aus8.value = ' ';
    document.MyForm.aus9.value = ' ';
    document.MyForm.aus9a.value = ' ';
    initCanvas(kb,2,1);
  }
```

```
  function Clear2() {
  // Grafik löschen
    document.MyForm.ein1.value = 0;
    document.MyForm.ein2.value = ' ';
    document.MyForm.aus0.value = ' ';
    document.MyForm.aus4.value = ' ';
    document.MyForm.aus5.value = ' ';
    document.MyForm.aus5a.value = ' ';
    document.MyForm.aus6.value = ' ';
    document.MyForm.aus6a.value = ' ';
    document.MyForm.aus6b.value = ' ';
    document.MyForm.aus8.value = ' ';
    document.MyForm.aus9.value = ' ';
    document.MyForm.aus9a.value = ' ';
    initCanvas(kb,2,1);
  }

function Calcul() {
// Hauptroutine
  kw = 1 * document.MyForm.ein0.value;
  initCanvas(kb,2,1);
  document.MyForm.aus0.value = ' ';
  document.MyForm.aus4.value = ' ';
  document.MyForm.aus5.value = ' ';
  document.MyForm.aus5a.value = ' ';
  document.MyForm.aus6.value = ' ';
  document.MyForm.aus6a.value = ' ';
  document.MyForm.aus6b.value = ' ';
  document.MyForm.aus8.value = ' ';
  document.MyForm.aus9.value = ' ';
  document.MyForm.aus9a.value = ' ';

  x = 1 * document.MyForm.ein1.value;
  s = document.MyForm.ein2.value;
  s = myTrim(s);
  y = fun(s,x);
  y = round2(y);
  document.MyForm.aus0.value = y;
  A.x = x;
  A.y = y;
  drawFun(kw,kb,s);                                    // ==> "graph.js"
  if (run < 1) { drawPoint('',A); }

  O.x = 0;
  O.y = 0;
  C.x = 0;
  C.y = 0;
  D.x = 0;
  D.y = 0;

  l = A.len();
  l = round2(l);
  w = A.win();
  w = round2(deg(w));
  // alert('Betrag OA = ' + l + ', Winkel OA = ' + w);

  x = 1 * document.MyForm.ein1.value;
  s = document.MyForm.ein2.value;
  ky = differ(s,x);                                     // ==> "mathe.js"
  kky = differ2(s,x)                                    // ==> "mathe.js"
//  ky = round2(ky);
  document.MyForm.aus5.value = round2(ky);
  document.MyForm.aus5a.value = round2(kky);

  if (tng) {
  drawWidth(1);
  drawColor("#000000");
  T = tangente(s,A);
  texT = ' y = ' + T.x + ' * x + (' + T.y + ')';
  if (Number.isNaN(T.y)) { texT = ' x = ' + T.x; }
  if (T.y == Infinity) { texT = ' x = ' + T.x; }

  document.MyForm.aus6.value = texT;

  if ( Number.isNaN(A.y) ) {
    A.y = 0;
    drawPoint("",A);
    document.MyForm.aus6.value = '';
  }
```

```
else {
if ( (Number.isNaN(T.y)) || (T.y == Infinity) ) {
  B.x = A.x;
  B.y = -0.999;
  //  drawPoint("B",B);
  drawLine1(A,B);
}
else {
  B.x = A.x - 0.999;
  B.y = T.x * B.x + T.y;
  //  drawPoint("B",B);
  drawLine1(A,B);
}
}
}

if (run == 1) {
// Nullstelle
tng = false;
x3 = prompt('Linke Intervallgrenze a','1');
x3 = myTrim(x3);
x3 = 1*x3;
x4 = prompt('Rechte Intervallgrenze b','3');
x4 = myTrim(x4);
x4 = 1*x4;
if (x3 == x4) { return; }
if (x4 < x3) { return; }

document.MyForm.aus6a.value = x3;
document.MyForm.aus6b.value = x4;
document.MyForm.aus0.value = '';
document.MyForm.aus5.value = '';
document.MyForm.aus5a.value = ' ';
document.MyForm.aus6.value = '';
s = document.MyForm.ein2.value;
xN = nullst(s,x3,x4);                                  // ==> "mathe.js"
if (xN == -999) {
   alert('Keine Nullstelle gefunden!');
   document.MyForm.ein1.value = '';
   return;
}
document.MyForm.ein1.value = xN;
xN = round2(xN);
document.MyForm.aus8.value = xN + '/0';
E.x = 1*xN;
E.y = 0;
drawPoint("N",E);
}

if (run == 2) {
// Extremstelle
tng = false;
x5 = prompt('Linke Intervallgrenze a','0');
x5 = myTrim(x5);
x5 = 1*x5;
x6 = prompt('Rechte Intervallgrenze b','3');
x6 = myTrim(x6);
x6 = 1*x6;
if (x5 == x6) { return; }
if (x6 < x5) { return; }

document.MyForm.aus6a.value = x5;
document.MyForm.aus6b.value = x6;
s = document.MyForm.ein2.value;
E = extrema(s,x5,x6);                                  // ==> "mathe.js"
document.MyForm.aus0.value = '';
document.MyForm.aus5.value = '';
document.MyForm.aus5a.value = ' ';
document.MyForm.aus6.value = '';
document.MyForm.ein1.value = E.x;
E.x = round2(E.x);
if (E.x == -999) {
  alert('Keine Extremstelle gefunden!');
  document.MyForm.ein1.value = '';
  return;
}
E.y = round2(E.y);
texE = ' ' + E.x + '/' + E.y;
document.MyForm.aus9.value = texE;
drawPoint("E",E);
}
```

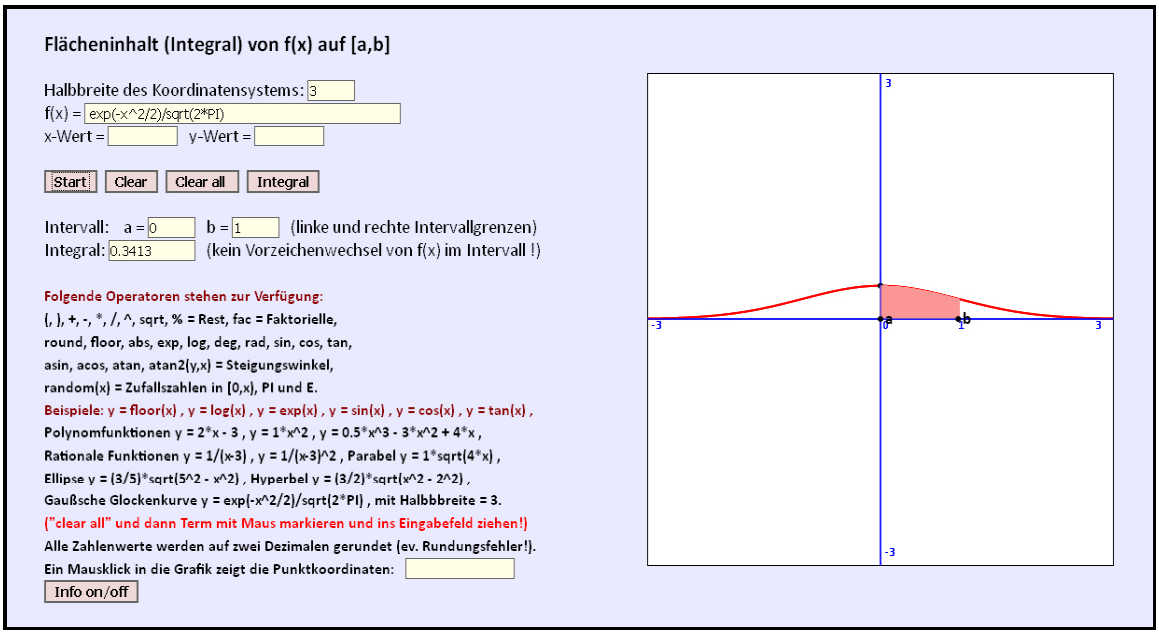
```
  if (run == 3) {
  // Wendestelle
  tng = false;
  x5 = prompt('Linke Intervallgrenze a','0');
  x5 = myTrim(x5);
  x5 = 1*x5;
  x6 = prompt('Rechte Intervallgrenze b','4');
  x6 = myTrim(x6);
  x6 = 1*x6;
  if (x5 == x6) { return; }
  if (x6 < x5) { return; }
  document.MyForm.aus6a.value = x5;
  document.MyForm.aus6b.value = x6;
  s = document.MyForm.ein2.value;
  W = wendpun(s,x5,x6);                                  // ==> "mathe.js"
  document.MyForm.aus0.value = '';
  document.MyForm.aus5.value = '';
  document.MyForm.aus5a.value = ' ';
  document.MyForm.aus6.value = '';
  document.MyForm.ein1.value = W.x;
  W.x = round2(W.x);
  if (W.x == -999) {
    alert('Keine Extremstelle gefunden!');
    document.MyForm.ein1.value = '';
    return;
  }
  W.y = round2(W.y);
  texW = ' ' + W.x + '/' + W.y;
  document.MyForm.aus9a.value = texW;
  drawPoint("W",W);
  }
 }

function Tang() {
  tng = true;
  Calcul();
  tng = false;
}
function Null() {
  run = 1;
  Calcul();
  run = 0;
}
function Extr() {
  run = 2;
  Calcul();
  run = 0;
}
function Wend() {
  run = 3;
  Calcul();
  run = 0;
}
</script>
</head>

<body onload = "initCanvas(kb,2,1)">

<div id='dc' style="position:absolute; top:70px; left:700px ">
  <canvas id='MyCanvas' width='500' height='500' onmousedown = "showPos(event)">
     Your browser does not support HTML5 Canvas.
  </canvas>
</div>

<form name="MyForm">
<h3>Kurvendiskussion von Funktionen</h3>
Halbbreite des Koordinatensystems: <input type="text" name="ein0" value="10" size="4" class="cd1">
<br>
Variable x = <input type="text" name="ein1" value="0" size="20" class="cd1">
<br>
f(x) = <input type="text" name="ein2" value="0.5*x^3 - 3*x^2 + 4*x" size="64" class="cd1">
<br>
y =     <input type="text" name="aus0" value=" " size="4" class="cd1"
                            readonly = "true">  
y' = <input type="text" name="aus5" value=" " size="4" class="cd1" readonly = "true">
  
y'' = <input type="text" name="aus5a" value=" " size="4" class="cd1" readonly = "true">
  
Tangente: <input type="text" name="aus6" value=" " size="29" class="cd1" readonly = "true">
<br>
<br>
```

```
<input type="button" name="calc1" value="Start" class="cd2" onclick="Calcul()"> 
<input type="button" name="calc2" value="Clear" class="cd2" onclick="Clear1()"> 
<input type="button" name="calc3" value="Clear all " class="cd2" onclick="Clear2()"> 
<input type="button" name="tang" value="Tangente" class="cd2" onclick="Tang()"> 
<input type="button" name="calc5" value="Nullstelle" class="cd2" onclick="Null()"> 
<input type="button" name="calc6" value="Extremum" class="cd2" onclick="Extr()"> 
<input type="button" name="calc7" value="Wende" class="cd2" onclick="Wend()"><br>
<br>

Intervall:     a = <input type="text" name="aus6a" value=" " size="5"
class="cd1" readonly = "true">  
b = <input type="text" name="aus6b" value=" " size="5" class="cd1" readonly = "true">
  (linke und rechte Intervallgrenzen)<br>
Nullstelle: <input type="text" name="aus8" value=" " size="11" class="cd1" readonly = "true">
  
Extremum: <input type="text" name="aus9" value=" " size="11" class="cd1" readonly = "true">
  
Wendestelle: <input type="text" name="aus9a" value=" " size="11" class="cd1" readonly = "true">
<br><br>

<b>
<font size="-0.80">
<font color="black">
Folgende Operatoren stehen zur Verfügung:<br>
 (, ), +, -, *, /, ^, sqrt, % = Rest, fac = Faktorielle, <br>
round, rd2, rd4, floor, abs, exp, log, lg, deg, rad, sin, cos, tan, <br>
asin, acos, atan, atan2(y,x) = Steigungswinkel,<br>
random(x) = Zufallszahlen in [0,x), PI und E.<br>
</font>
<font color = "darkred">
Beispiele: y = floor(x) , y = log(x) , y = exp(x) , y = sin(x) , y = cos(x) , y = tan(x) ,<br>
Polynomfunktionen y = 2*x - 3 , y = 1*x^2 , y = 0.5*x^3 - 3*x^2 + 4*x ,<br>
Rationale Funktionen y = 1/(x-3) , y = x^3/(x^2-4) , Parabel y = 1*sqrt(4*x) ,<br>
Ellipse y = (4/6)*sqrt(6^2 - x^2) , Hyperbel y = (3/2)*sqrt(x^2 - 2^2).<br>
Gaußsche Glockenkurve y = exp(-x^2/2)/sqrt(2*PI) , mit Halbbreite = 3.<br>
</font>
<font color = "red">
("clear all" und dann Term mit Maus markieren und ins Eingabefeld ziehen!)<br>
</font>
<font color = "black">
Alle Zahlenwerte werden auf zwei Dezimalen gerundet (ev. Rundungsfehler!).<br>
Ein Mausklick in die Grafik zeigt die Punktkoordinaten:  
<input type="text" name="aus4" value=" " size="16" class="cd1" readonly><br>
</font>
</font>
</b>

</form>

<script>
   function showPos(ev) {
   kw = 1 * document.MyForm.ein0.value;
   x1 = ev.clientX - 700;
   y1 = ev.clientY - 70;
   x0 = x1*2*kw/kb - kw;
   y0 = (kb - y1)*2*kw/kb - kw;
   info = round2(x0) + '  /  ' + round2(y0);
   document.MyForm.aus4.value = info;
   }
</script>

</body>
</html>
```

[4.3] Integralrechnung

[4.3.1] Das Programm *„integral1.html“*

```
-------------------------------------------------------------------------
Mit dem bestimmten Integral ∫f(x)dx einer stetigen Funktion f(x)
auf dem Intervall [a,b] wird jene Fläche in der Ebene ermittelt,
welche von der Funktionskurve y = f(x), der x-Achse, und links
und rechts von den beiden Geraden x = a und x = b begrenzt wird.
-------------------------------------------------------------------------
```




```
<!DOCTYPE html>
<html>
<head>
<title>Integralrechnung</title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Integralrechnung">

<style type = "text/css">
body {background-color:#EAEAFF; color:black; font-family:Calibri,Arial;
      font-size:19px; font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
.cd1 {border: 1px solid #666666; background-color: #FFFFE8; font-size: 14px;}
.cd2 {border: 2px solid #666666; background-color: #EED8D8; font-size: 14px; font-weight: bold;}
</style>

<script src="parser.js"></script>
<script src="graph.js"></script>
<script src="mathe.js"></script>

<script>

  var run = 0;
  var s;  // Funktionsterm
  var fla = 0;
  var hide = true;

  A = new TPoint(0,0);
  B = new TPoint(0,0);
  C = new TPoint(0,0);
  D = new TPoint(0,0);
  E = new TPoint(0,0);
  O = new TPoint(0,0);
```

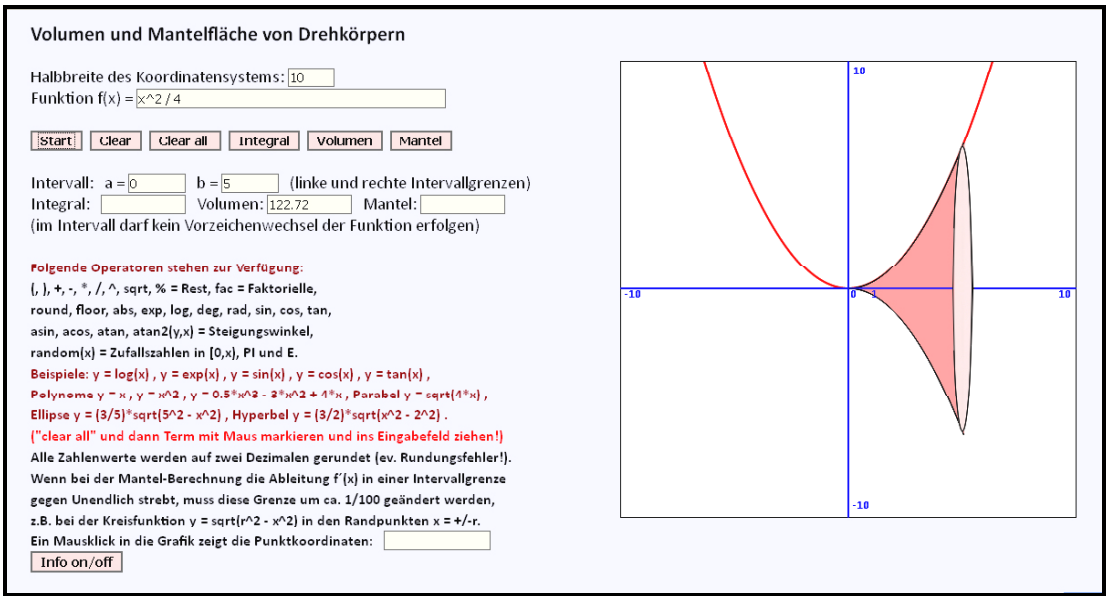
```
  function Clear1() {
  // Grafik löschen
    document.MyForm.ein1.value = 0;
    document.MyForm.aus0.value = ' ';
    document.MyForm.aus4.value = ' ';
    document.MyForm.aus6a.value = ' ';
    document.MyForm.aus6b.value = ' ';
    document.MyForm.aus7.value = ' ';
    initCanvas(kb,2,1);
    document.MyForm.calc1.focus();
  }

  function Clear2() {
  // Grafik löschen
    document.MyForm.ein1.value = 0;
    document.MyForm.ein2.value = ' ';
    document.MyForm.aus0.value = ' ';
    document.MyForm.aus4.value = ' ';
    document.MyForm.aus6a.value = ' ';
    document.MyForm.aus6b.value = ' ';
    document.MyForm.aus7.value = ' ';
    initCanvas(kb,2,1);
    document.MyForm.calc1.focus();
  }

function Calcul() {
// Hauptroutine
  kw = 1 * document.MyForm.ein0.value;
  initCanvas(kb,2,1);
  document.MyForm.aus0.value = ' ';
  document.MyForm.aus4.value = ' ';
  document.MyForm.aus6a.value = ' ';
  document.MyForm.aus6b.value = ' ';
  document.MyForm.aus7.value = ' ';
  x = 1 * document.MyForm.ein1.value;
  s = document.MyForm.ein2.value;
  s = myTrim(s);
  y = fun(s,x);
  y = round4(y);
  document.MyForm.aus0.value = y;
  A.x = x;
  A.y = y;
  drawFun(kw,kb,s);                                    // ==> "graph.js"
  drawPoint('',A);
  O.x = 0;
  O.y = 0;
  C.x = 0;
  C.y = 0;
  D.x = 0;
  D.y = 0;
  l = A.len();
  l = round4(l);
  w = A.win();
  w = round4(deg(w));

  if (run == 1) {
  document.MyForm.ein1.value = '';
  document.MyForm.aus0.value = '';
  x1 = prompt('Linke Intervallgrenze a','0');
  x1 = myTrim(x1);
  x1 = 1*x1;
  C.x = x1; C.y = 0;
  document.MyForm.aus6a.value = x1;
  x2 = prompt('Rechte Intervallgrenze b','2');;
  x2 = myTrim(x2);
  x2 = 1*x2;
  if (x1 == x2) { return; }
  if (x2 < x1) { return; }

  s = document.MyForm.ein2.value;
  D.x = x2; D.y = 0;
  document.MyForm.aus6b.value = x2;
  drawWidth(2);
  drawColor("rgb(250,150,150)");
  fillFun1(kw,kb,x1,x2,s);
  drawPoint("a",C);
  drawPoint("b",D);
```

```
  fla = integ(s,x1,x2);                                    // ==> "mathe.js"
  if (fla == -999) {
      alert('Undefinierte Integrationsstellen!');
      return;
  }
  fla = round4(fla);
  document.MyForm.aus7.value = fla;
  }
  document.MyForm.calc1.focus();
 }

function Inte() {
  run = 1;
  Calcul();
  run = 0;
}
function Info(){
  hide = !hide;
  if (hide) {
    document.getElementById("dc").style.visibility = 'visible';
    document.getElementById("cd3").style.visibility = 'hidden';
  }
  if (!hide) {
    document.getElementById("dc").style.visibility = 'hidden';
    document.getElementById("cd3").style.visibility = 'visible';
  }
}
</script>
</head>

<body onload = "initCanvas(kb,2,1)">
<div id='dc' style="position:absolute; top:70px; left:700px ">
  <canvas id='MyCanvas' width='500' height='500' onmousedown = "showPos(event)">
     Your browser does not support HTML5 Canvas.
  </canvas>
</div>

<form name="MyForm">
<h3>Flächeninhalt (Integral) von f(x) auf [a,b]</h3>
Halbbreite des Koordinatensystems: <input type="text" name="ein0" value="5" size="4" class="cd1"><br>
f(x) = <input type="text" name="ein2" value="0.6*x^3 - 3*x^2 + 4*x" size="52" class="cd1"><br>
x-Wert = <input type="text" name="ein1" value="0" size="8" class="cd1">  
y-Wert = <input type="text" name="aus0" value=" " size="8" class="cd1" readonly = "true"><br>
<br>
<input type="button" name="calc1" value="Start" class="cd2" onclick="Calcul()"> 
<input type="button" name="calc2" value="Clear" class="cd2" onclick="Clear1()"> 
<input type="button" name="calc3" value="Clear all " class="cd2" onclick="Clear2()"> 
<input type="button" name="calc4" value="Integral" class="cd2" onclick="Inte()"><br>
<br>
Intervall:    a = <input type="text" name="aus6a" value=" " size="4" class="cd1"
readonly = "true">  
b = <input type="text" name="aus6b" value=" " size="4" class="cd1" readonly = "true">  
(linke und rechte Intervallgrenzen)<br>
Integral: <input type="text" name="aus7" value=" " size="11" class="cd1" readonly = "true">
   (kein Vorzeichenwechsel von f(x) im Intervall !)<br>
<br>
<b>
<font size="-0.80">
<font color="darkred">
Folgende Operatoren stehen zur Verfügung:<br>
</font>
<font color = "black">
(, ), +, -, *, /, ^, sqrt, % = Rest, fac = Faktorielle, <br>
round, rd2, rd4, floor, abs, exp, log, lg, deg, rad, sin, cos, tan, <br>
asin, acos, atan, atan2(y,x) = Steigungswinkel,<br>
random(x) = Zufallszahlen in [0,x), PI und E.<br>
</font>
<font color = "darkred">
Beispiele: y = floor(x) , y = log(x) , y = exp(x) , y = sin(x) , y = cos(x) , y = tan(x) ,<br>
</font>
Polynomfunktionen y = 2*x - 3 , y = 1*x^2 , y = 0.5*x^3 - 3*x^2 + 4*x ,<br>
Rationale Funktionen y = 1/(x-3) , y = 1/(x-3)^2 , Parabel y = 1*sqrt(4*x) ,<br>
Ellipse y = (3/5)*sqrt(5^2 - x^2) , Hyperbel y = (3/2)*sqrt(x^2 - 2^2) ,<br>
Gaußsche Glockenkurve y = exp(-x^2/2)/sqrt(2*PI) , mit Halbbreite = 3.<br>
<font color = "red">
("clear all" und dann Term mit Maus markieren und ins Eingabefeld ziehen!)<br>
</font>
```

```
<font color = "black">
Alle Zahlenwerte werden auf zwei Dezimalen gerundet (ev. Rundungsfehler!).<br>
Ein Mausklick in die Grafik zeigt die Punktkoordinaten:  
<input type="text" name="aus4" value=" " size="15" class="cd1" readonly><br>
<input type="button" name="btn" value="Info on/off" class="cd2" onclick="Info()"><br>
</b>
</font>
</font>
</form>

<div id='cd3' style="position:absolute; top:70px; left:700px; visibility:hidden;">
<font color = "black">
<br><br>
-------------------------------------------------------------------------------------<br>
Mit dem bestimmten Integral &int;f(x)dx einer stetigen Funktion f(x)<br>
auf dem Intervall [a,b] wird jene Fläche in der Ebene ermittelt,<br>
die von der der Funktionskurve y = f(x), der x-Achse, und links<br>
und rechts von den beiden Geraden x = a und x = b begrenzt wird.<br>
-------------------------------------------------------------------------------------<br>
<br>
</font>
</div>

<script>
   function showPos(ev) {
   kw = 1 * document.MyForm.ein0.value;
   x3 = ev.clientX - 700;
   y3 = ev.clientY - 70;
   x4 = x3*2*kw/kb - kw;
   y4 = (kb - y3)*2*kw/kb - kw;
   info = round2(x4) + '  /  ' + round2(y4);
   document.MyForm.aus4.value = info;
   }
   document.MyForm.calc1.focus();
</script>
</body>
</html>
```

[4.3.2] Das Programm *„integral2.html“*

```
-------------------------------------------------------------------------
Gegeben ist eine stetige Funktion y = f(x) auf dem Intervall [a,b].
Die Fläche A unter der Kurve wird mit dem Integral ∫ydx bestimmt.
Die Kurve rotiert um die x-Achse und erzeugt dabei einen Drehkörper.
Für das Volumen V des Drehkörpers gilt: V = ¶*∫y²dx.
Für die Mantelfläche M des Körpers gilt: M = 2¶*∫y*sqrt(1+[y']²)dx.
y' = f'(x) ist die Ableitung der Funktion, sqrt ist die Quadratwurzel.
-------------------------------------------------------------------------
```




```
<!DOCTYPE html>
<html>
<head>
<title>Drehkörper, Volumen und Mantel</title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Drehkörper, Volumen und Mantel">

<style type = "text/css">
body {background-color:#EAEAFF; color:black; font-family:Calibri,Arial;
      font-size:19px; font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
.cd1 {border: 1px solid #666666; background-color: #FFFFE8; font-size: 14px;}
.cd2 {border: 2px solid #666666; background-color: #EED8D8; font-size: 14px; font-weight: bold;}
</style>

<script src="parser.js"></script>
<script src="graph.js"></script>
<script src="mathe.js"></script>

<script>
  var run = 0;
  var hide = true;

  A = new TPoint(0,0);
  B = new TPoint(0,0);
  C = new TPoint(0,0);
  D = new TPoint(0,0);
  E = new TPoint(0,0);
  F = new TPoint(0,0);
  O = new TPoint(0,0);
  G = new TPoint(0,0);
  H = new TPoint(0,0);

  var s;  // +Funktionsterm
  var t;  // -Funktionsterm
  var x1 = 0;
  var x2 = 5;
  var fla;
```

```
  function Clear1() {
  // Grafik löschen
    document.MyForm.aus4.value = ' ';
    document.MyForm.aus6a.value = ' ';
    document.MyForm.aus6b.value = ' ';
    document.MyForm.aus7.value = ' ';
    document.MyForm.aus8.value = ' ';
    document.MyForm.aus9.value = ' ';
    initCanvas(kb,2,1);
    document.MyForm.calc1.focus();
  }

  function Clear2() {
  // Grafik löschen
    x1 = 0;
    x2 = 5;
    document.MyForm.ein2.value = ' ';
    document.MyForm.aus4.value = ' ';
    document.MyForm.aus6a.value = ' ';
    document.MyForm.aus6b.value = ' ';
    document.MyForm.aus7.value = ' ';
    document.MyForm.aus8.value = ' ';
    document.MyForm.aus9.value = ' ';
    initCanvas(kb,2,1);
    document.MyForm.calc1.focus();
  }

function Calcul() {
// Hauptroutine
  kw = 1 * document.MyForm.ein0.value;
  initCanvas(kb,2,1);
  document.MyForm.aus4.value = ' ';
  document.MyForm.aus6a.value = ' ';
  document.MyForm.aus6b.value = ' ';
  document.MyForm.aus7.value = ' ';
  document.MyForm.aus8.value = ' ';
  document.MyForm.aus9.value = ' ';
  s = document.MyForm.ein2.value;
  s = myTrim(s);
  t = '-(' + s + ')';
  drawFun(kw,kb,s);                                    // ==> "graph.js"
  O.x = 0;
  O.y = 0;
  C.x = 0;
  C.y = 0;
  D.x = 0;
  D.y = 0;
  E.x = 0;
  E.y = 0;

  if (run == 1) {
// Integral
  s = document.MyForm.ein2.value;
  s = myTrim(s);
  x1 = prompt('Linke Intervallgrenze a',x1);
  x1 = myTrim(x1);
  x1 = 1*x1;
  s = document.MyForm.ein2.value;
  s = myTrim(s);
  y2 = fun(s,x1);
  C.x = x1; C.y = 0;
  E.x = x1; E.y = y2;
  x2 = prompt('Rechte Intervallgrenze b',x2);;
  x2 = myTrim(x2);
  x2 = 1*x2;
  if (x1 == x2) { return; }
  if (x2 < x1) { return; }

  s = document.MyForm.ein2.value;
  s = myTrim(s);
  y2 = fun(s,x2);
  D.x = x2; D.y = 0;
  F.x = x2; F.y = y2;
  document.MyForm.aus6a.value = x1;
  document.MyForm.aus6b.value = x2;
  drawWidth(2);
  drawColor("rgb(250,150,150)");
  fillFun1(kw,kb,x1,x2,s);
  drawPoint("a",C);
  drawPoint("b",D);
```

```
fla = integ(s,x1,x2);                                    // ==> "mathe.js"
if (fla == -999) {
   alert('Abbruch: f(x) = Undefined oder f´(x) = Infinity!');
   return;
}
fla = round2(fla);
document.MyForm.aus7.value = fla;
}

if (run == 2) {
// Volumen
s = document.MyForm.ein2.value;
s = myTrim(s);
x1 = prompt('Linke Intervallgrenze a',x1);
x1 = myTrim(x1);
x1 = 1*x1;
s = document.MyForm.ein2.value;
s = myTrim(s);
y1 = fun(s,x1);
C.x = x1; C.y = 0;
E.x = x1; E.y = y1;
x2 = prompt('Rechte Intervallgrenze b',x2);;
x2 = myTrim(x2);
x2 = 1*x2;
if (x1 == x2) { return; }
if (x2 < x1) { return; }

s = document.MyForm.ein2.value;
s = myTrim(s);
y2 = fun(s,x2);
D.x = x2; D.y = 0;
F.x = x2; F.y = y2;
document.MyForm.aus6a.value = x1;
document.MyForm.aus6b.value = x2;
drawWidth(2);
drawColor("rgb(250,150,150)");
fillFun1(kw,kb,x1,x2,s);
drawFun1(kw,kb,x1,x2,s);                                 // ==> "graph.js"
drawColor("rgb(250,150,150)");
fillFun1(kw,kb,x1,x2,t);
drawFun1(kw,kb,x1,x2,t);

fla = integvol(s,x1,x2);                                 // ==> "mathe.js"
if (fla == -999) {
   alert('Abbruch: f(x) = Undefined oder f´(x) = Infinity!');
   return;
}
fla = round2(fla);
document.MyForm.aus8.value = fla;

G.x = E.x; G.y = -E.y;
H.x = F.x; H.y = -F.y;
drawLine(E,G);
drawLine(F,H);
f = 1.5 * document.MyForm.ein0.value;

if ( Math.abs(y1) > Math.abs(y2) ) {
   drawColor("rgb(240,220,220)");
   fillEll(C,y1/f,y1,1);
   fillEll(C,y1/f,y1,-1);
   drawColor("rgb(0,0,0)");
   drawEll(C,y1/f,y1,1);
   drawEll(C,y1/f,y1,-1);
}
else {
   drawColor("rgb(240,220,220)");
   fillEll(D,y2/f,y2,1);
   fillEll(D,y2/f,y2,-1);
   drawColor("rgb(0,0,0)");
   drawEll(D,y2/f,y2,1);
   drawEll(D,y2/f,y2,-1);
}
}
```

```
  if (run == 3) {
  // Oberfläche
  s = document.MyForm.ein2.value;
  s = myTrim(s);
  x1 = prompt('Linke Intervallgrenze a',x1);
  x1 = myTrim(x1);
  x1 = 1*x1;
  y1 = fun(s,x1);
  C.x = x1; C.y = 0;
  E.x = x1; E.y = y1;
  x2 = prompt('Rechte Intervallgrenze b',x2);;
  x2 = myTrim(x2);
  x2 = 1*x2;
  if (x1 == x2) { return; }
  if (x2 < x1) { return; }
  y2 = fun(s,x2);
  D.x = x2; D.y = 0;
  F.x = x2; F.y = y2;
  document.MyForm.aus6a.value = x1;
  document.MyForm.aus6b.value = x2;
  drawWidth(2);
  drawColor("rgb(250,150,150)");
  fillFun1(kw,kb,x1,x2,s);
  drawFun1(kw,kb,x1,x2,s);                                  // ==> "graph.js"
  drawColor("rgb(250,150,150)");
  fillFun1(kw,kb,x1,x2,t);
  drawFun1(kw,kb,x1,x2,t);

  fla = integvol(s,x1,x2);                                  // ==> "mathe.js"
  if (fla == -999) {
     alert('Abbruch: f(x) = Undefined oder f´(x) = Infinity!');
     return;
  }
  fla = round2(fla);
  document.MyForm.aus8.value = fla;


  fla = integfla(s,x1,x2);                                  // ==> "mathe.js"
  if (fla == -999) {
     alert('Abbruch: f(x) = Undefined oder f´(x) = Infinity!');
     return;
  }
  fla = round2(fla);
  document.MyForm.aus9.value = fla;

  G.x = E.x; G.y = -E.y;
  H.x = F.x; H.y = -F.y;
  drawLine(E,G);
  drawLine(F,H);
  f = 1.5 * document.MyForm.ein0.value;
  if ( Math.abs(y1) > Math.abs(y2) ) {
     drawColor("rgb(240,220,220)");
     fillEll(C,y1/f,y1,1);
     fillEll(C,y1/f,y1,-1);
     drawColor("rgb(0,0,0)");
     drawEll(C,y1/f,y1,1);
     drawEll(C,y1/f,y1,-1);
  }
  else {
     drawColor("rgb(240,220,220)");
     fillEll(D,y2/f,y2,1);
     fillEll(D,y2/f,y2,-1);
     drawColor("rgb(0,0,0)");
     drawEll(D,y2/f,y2,1);
     drawEll(D,y2/f,y2,-1);
  }
//  drawPoint("a",C);
//  drawPoint("b",D);
  }
  document.MyForm.calc1.focus();
 }

function Inte() {
  run = 1;
  Calcul();
  run = 0;
}

function Volu() {
  run = 2;
  Calcul();
  run = 0;
}
```

```
function Flae() {
  run = 3;
  Calcul();
  run = 0;
}

function Info(){
  hide = !hide;
  if (hide) {
    document.getElementById("dc").style.visibility = 'visible';
    document.getElementById("cd3").style.visibility = 'hidden';
  }
  if (!hide) {
    document.getElementById("dc").style.visibility = 'hidden';
    document.getElementById("cd3").style.visibility = 'visible';
  }
}

</script>
</head>

<body onload = "initCanvas(kb,2,1)">
<div id='dc' style="position:absolute; top:70px; left:700px ">
  <canvas id='MyCanvas' width='500' height='500' onmousedown = "showPos(event)">
     Your browser does not support HTML5 Canvas.
  </canvas>
</div>

<form name="MyForm">
<h3>Volumen und Mantelfläche von Drehkörpern</h3>
Halbbreite des Koordinatensystems: <input type="text" name="ein0" value="10" size="4"
class="cd1"><br>
Funktion f(x) = <input type="text" name="ein2" value="x^2 / 4" size="52" class="cd1"><br>
<br>
<input type="button" name="calc1" value="Start" class="cd2" onclick="Calcul()"> 
<input type="button" name="calc2" value="Clear" class="cd2" onclick="Clear1()"> 
<input type="button" name="calc3" value="Clear all " class="cd2" onclick="Clear2()"> 
<input type="button" name="calc4" value="Integral" class="cd2" onclick="Inte()"> 
<input type="button" name="calc5" value="Volumen" class="cd2" onclick="Volu()"> 
<input type="button" name="calc6" value="Mantel" class="cd2" onclick="Flae()"><br>
<br>
Intervall:   a = <input type="text" name="aus6a" value=" " size="6" class="cd1"
                           readonly = "true">  
b = <input type="text" name="aus6b" value=" " size="6" class="cd1" readonly = "true">
   (linke und rechte Intervallgrenzen)<br>
Integral:   <input type="text" name="aus7" value=" " size="11" class="cd1"
                       readonly = "true">  
Volumen: <input type="text" name="aus8" value=" " size="11" class="cd1"
                       readonly = "true">  
Mantel: <input type="text" name="aus9" value=" " size="11" class="cd1" readonly = "true"><br>
(im Intervall darf kein Vorzeichenwechsel der Funktion erfolgen)<br>
<br>
<b>
<font size="-0.80">
<font color="darkred">
Folgende Operatoren stehen zur Verfügung:<br>
</font>
<font color = "black">
(, ), +, -, *, /, ^, sqrt, % = Rest, fac = Faktorielle, <br>
round, rd2, rd4, floor, abs, exp, log, lg, deg, rad, sin, cos, tan, <br>
asin, acos, atan, atan2(y,x) = Steigungswinkel,<br>
random(x) = Zufallszahlen in [0,x), PI und E.<br>
</font>
<font color = "darkred">
Beispiele: y = log(x) , y = exp(x) , y = sin(x) , y = cos(x) , y = tan(x) ,<br>
Polynome y = x , y = x^2 , y = 0.5*x^3 - 3*x^2 + 4*x , Parabel y = sqrt(4*x) ,<br>
Ellipse y = (3/5)*sqrt(5^2 - x^2) , Hyperbel y = (3/2)*sqrt(x^2 - 2^2) .<br>
</font>
<font color = "red">
("clear all" und dann Term mit Maus markieren und ins Eingabefeld ziehen!)<br>
</font>
<font color = "black">
Alle Zahlenwerte werden auf zwei Dezimalen gerundet (ev. Rundungsfehler!).<br>
Wenn bei der Mantel-Berechnung die Ableitung f´(x) in einer Intervallgrenze<br>
gegen Unendlich strebt, muss diese Grenze um ca. 1/100 geändert werden,<br>
z.B. bei der Kreisfunktion y = sqrt(r^2 - x^2) in den Randpunkten x = +/-r.<br>
Ein Mausklick in die Grafik zeigt die Punktkoordinaten:  
<input type="text" name="aus4" value=" " size="15" class="cd1" readonly><br>
<input type="button" name="btn" value="Info on/off" class="cd2" onclick="Info()"><br>
</b>
</font>
</font>
</form>
```

```
<div id='cd3' style="position:absolute; top:70px; left:700px; visibility:hidden;">
<font color = "black">
<br><br>
-------------------------------------------------------------------------------------------<br>
Gegeben ist eine stetige Funktion y = f(x) auf dem Intervall [a,b].<br>
Die Fläche A unter der Kurve wird mit dem Integral &int;ydx bestimmt.<br>
Die Kurve rotiert um die x-Achse und erzeugt dabei einen Drehkörper.<br>
Für das Volumen V des Drehkörpers gilt: V = ¶*&int;y²dx.<br>
Für die Mantelfläche M des Körpers gilt: M = 2¶*&int;y*sqrt(1+[y']²)dx.<br>
y' = f'(x) ist die Ableitung der Funktion, sqrt ist die Quadratwurzel.<br>
-------------------------------------------------------------------------------------------<br>
<br>
</font>
</div>

<script>
   function showPos(ev) {
   kw = 1 * document.MyForm.ein0.value;
   x3 = ev.clientX - 700;
   y3 = ev.clientY - 70;
   x4 = x3*2*kw/kb - kw;
   y4 = (kb - y3)*2*kw/kb - kw;
   info = round2(x4) + '  /  ' + round2(y4);
   document.MyForm.aus4.value = info;
   }
   document.MyForm.calc1.focus();
</script>
</body>
</html>
```

[4.4] Die JavaScript-Datei *„mathe.js“*

```
// ------------------------------------------------------------------------
// "mathe.js", Sammlung von Mathematik-Routinen
// (c) Herbert Paukert, Version 4.0 am 10.12.2018
// ------------------------------------------------------------------------

function myTrim(x) { return x.replace(/^\s+|\s+$/gm,'') }
function round2(x) { return Math.round(100*x)/100 }
function round4(x) { return Math.round(10000*x)/10000 }
function deg(x) { return x * 180 / Math.PI; }
function rad(x) { return x * Math.PI / 180; }

var info, info1, info2, info3, info4;
var error = -999;

n = 3;
// Erster Lösungsvektor
var L1xy = new Array(n);
L1xy[0] = 0;
L1xy[1] = 0;
L1xy[2] = 0;
L1xy[3] = 0;

// Zweiter Lösungsvektor
var L2xy = new Array(n);
L2xy[0] = 0;
L2xy[1] = 0;
L2xy[2] = 0;
L2xy[3] = 0;

// mehrdimensionale (n)x(n+1)-Matrix
var mat = new Array(n);
for (var i = 0; i <= n+1; i++) { mat[i]= new Array(n+1); }

function TPoint(tx, ty) {
  // Konstruktor für das Punkt-Objekt (zweidimensional)
  this.x = tx;
  this.y = ty;
  this.len = function() {
    var z = Math.sqrt(this.x*this.x + this.y*this.y); return round2(z);
  }
  this.win = function() {
    if ((this.x < 0) && (this.y == 0)) { z = 180; return z;}
    if ((this.x == 0) && (this.y < 0)) { z = 270; return z; }
    if ((this.x == 0) && (this.y == 0)) { z = 0; return z; }
    var z = Math.atan(this.y/this.x);
    if (z == Infinity) { z = 90; return z; }
    z = deg(Math.abs(z));
    if ((this.x < 0) && (this.y > 0)) { z = 180 - z; }
    if ((this.x < 0) && (this.y < 0)) { z = 180 + z; }
    if ((this.x > 0) && (this.y < 0)) { z = 360 - z; }
    if (z >= 360 ) {z = z - 360; }
    return round2(z);
  }
}

function TPoint3(tx, ty, tz) {
  // Konstruktor für das Punkt-Objekt (dreidimensional)
  this.x = tx;
  this.y = ty;
  this.z = tz;
  this.len = function() {
    var z = Math.sqrt(this.x*this.x + this.y*this.y + this.z*this.z); return round2(z);
  }
  this.win = function() {
   if ((this.x < 0) && (this.y == 0)) { z = 180; return z;}
    if ((this.x == 0) && (this.y < 0)) { z = 270; return z; }
    if ((this.x == 0) && (this.y == 0)) { z = 0; return z; }
    var z = Math.atan(this.y/this.x);
    if (z == Infinity) { z = 90; return z; }
    z = deg(Math.abs(z));
    if ((this.x < 0) && (this.y > 0)) { z = 180 - z; }
    if ((this.x < 0) && (this.y < 0)) { z = 180 + z; }
    if ((this.x > 0) && (this.y < 0)) { z = 360 - z; }
    if (z >= 360 ) {z = z - 360; }
    return round2(z);
  }
}
```

```
function Max6(a,b,c,d,e,f) {
  // Maximum von 6 Zahlen
  var z = a;
  if (b > z) { z = b; }
  if (c > z) { z = c; }
  if (d > z) { z = d; }
  if (e > z) { z = e; }
  if (f > z) { z = f; }
  return z;
}

function Min6(a,b,c,d,e,f) {
  // Minimum von 6 Zahlen
  var z = a;
  if (b < z) { z = b; }
  if (c < z) { z = c; }
  if (d < z) { z = d; }
  if (e < z) { z = e; }
  if (f < z) { z = f; }
  return z;
}

function Flaeche1(a,b,c) {
  // Fläche eines Dreiecks
  var u1,s1,erg;
  u1 = (a + b + c) / 2;
  s1 = u1*(u1-a)*(u1-b)*(u1-c);
  erg = Math.sqrt(s1);
  return erg;
}

function Winkel1(a,b,c) {
  // Winkel w bei Eckpunkt B
  var x1,y1,z1;
  x1 = (a*a + c*c - b*b) / (2*a*c);
  y1 = Math.acos(x1);
  z1 = deg(y1);
  return z1;
}

function ggt(a,b) {
// größter gemeinsamer Teiler
  var x,y,r;
  x = a;
  y = b;
  do {
    r = x % y;
    x = y;
    y = r;
  }
  while (r > 0);
  return x;
}

function kgv(a,b) {
// kleinstes gemeinsames Vielfache
var x,y;
x = ggt(a,b);
y = a * b / x;
return y;
}

function Primfakt(x) {
// Primfaktoren einer Zahl
var zahlen = new Array(1000);
var n,m,t;
function Init() {
  for (var i = 0; i < 1000; i++) { zahlen[i] = 0; }
}
function ausgabe(j) {
  var aus = '';
  for (var i = 1; i < j; i++) { aus = aus + zahlen[i] + '*'; }
  aus = aus.substring(0,aus.length-1);
  return aus;
}
function Teiler() {
while (((x % t) == 0) && (x > 1)) {
  n++;
  zahlen[n] = t;
  x = (x / t);
}
}
```

```
Init();
n = 0;
if (x < 0) { x = -x; }
m = x / 2;
t = 2;
Teiler();
t = 3;
do {
  Teiler();
  t = 1*t + 2;
}
while (t <= m);
if (n == 0) { n = 1; zahlen[n] = x; }
return ausgabe(n+1);
}

function differ(w9,x9) {
  // numerisches Differenzieren, w9 = Funktionsterm f(x), x9 = Argument, 1.Ableitung = y9
  var h9 = 0.0001;
  var z9 = fun(w9,x9);
  if (z9 == Infinity) { return z9; }
  var y9 =
    (fun(w9,x9-2*h9)-8*fun(w9,x9-h9)+8*fun(w9,1*x9+h9)-fun(w9,1*x9+2*h9))/(12*h9);
  return y9;
}

function differ2(w9,x9) {
  // numerisches Differenzieren, w9 = Funktionsterm f(x), x9 = Argument, 2.Ableitung = y9
  var h9 = 0.0001;
  var z9 = differ(w9,x9);
  if (z9 == Infinity) { return z9; }
  var y9 =
    (differ(w9,x9-2*h9)-8*differ(w9,x9-h9)+8*differ(w9,1*x9+h9)-differ(w9,1*x9+2*h9))/(12*h9);
  return y9;
}

function tangente(u9,Q9) {
  // Tangente y = k*x + d in Q9 und u9 als f(x), ergibt: k = Q1.x und d = Q1.y
  var Q1 = new TPoint;
  var a, b, c, k, d;
  a = Q9.x;
  b = fun(u9,a);
  if (b == Infinity) { Q1.x = a; Q1.y = b; return Q1 }
  k = differ(u9,a);
  if ( Number.isNaN(k) ) { Q1.x = a; Q1.y = k; return Q1 }
  d = b - k*a;
  Q1.x = round2(k);
  Q1.y = round2(d);
  return Q1;
}

function integ(u,a,b) {
   // bestimmtes Integral von a bis b (mit u als f(x))
   var x,y,d,sum;
   var x0,y0,p,rest;
   var n,i,erg;
   n = 1000;
   d = (b-a)/n;
   sum = fun(u,a);
   if (Number.isNaN(sum)) { erg = -999; return erg; }
   y = fun(u,b);
   if (Number.isNaN(y)) { erg = -999; return erg; }
   sum = sum + y;
   for (i=1; i <= (n-1); i++) {
     x = a + d*i;
     x0 = x-d;
     y = fun(u,x);
     y0 = fun(u,x0);
     p = y0 * y;
     if (p < 0) { erg = 0; return erg; }
     rest = i - Math.floor(i/2)*2;
     if (rest == 1) { sum = 1*sum + 4*y }
     else { sum = 1*sum + 2*y }
   }
   erg = sum*d/3;
   return erg;
}
```

```
function nullst(u,a,b) {
   // Nullstelle zwischen a und b (mit u als f(x))
   var gen = 0.0001;
   var i,maxi;
   var m,ya,ym,prod;
   var erg;
   erg = 0;
   maxi = 1000;
   i = 1;
   do {
     i = i + 1;
     if (i > maxi) { erg = -999; return erg; }
     ya = fun(u,a);
     if (Number.isNaN(ya)) { erg = -999; return erg; }
     if (Math.abs(ya) < gen) { erg = a; return erg; }
     m = (1*a+1*b)/2;
     ym = fun(u,m);
     if (Number.isNaN(ym)) { erg = -999; return erg; }
     prod = ya * ym;
     if (prod > 0) { a = m; }
     else { b = m; }
   } while (Math.abs(ym) > gen);
   erg = m;
   return erg;
}

function extrema(u,a,b) {
   // Extremstelle zwischen a und b (mit u als f(x))
   var gen = 0.0001;
   var i,maxi;
   var m,ya,ym,prod;
   var ya0,ym0;
   var Q1 = new TPoint;
   maxi = 1000;
   i = 1;
   do {
     i = i + 1;
     if (i > maxi) { Q1.x = -999; Q1.y = 0; return Q1;}
     ya0 = fun(u,a);
     if (Number.isNaN(ya0)) { erg = -999; return erg; }
     ya = differ(u,a);
     if (Math.abs(ya) < gen) { Q1.x = a; Q1.y = ya; return Q1; }
     m = (1*a+1*b)/2;
     ym0 = fun(u,m);
     if (Number.isNaN(ym0)) { erg = -999; return erg; }
     ym = differ(u,m);
     prod = ya* ym;
     if (prod > 0) { a = m; }
     else { b = m; }
   } while (Math.abs(ym) > gen);
     Q1.x = m;
     Q1.y = ym0;
     return Q1;
}

function wendpun(u,a,b) {
   // Wendestelle zwischen a und b (mit u als f(x))
   var gen = 0.0001;
   var i,maxi;
   var m,ya,ym,prod;
   var ya0,ym0;
   var Q1 = new TPoint;
   maxi = 1000;
   i = 1;
   do {
     i = i + 1;
     if (i > maxi) { Q1.x = -999; Q1.y = 0; return Q1;}
     ya0 = fun(u,a);
     if (Number.isNaN(ya0)) { erg = -999; return erg; }
     ya = differ2(u,a);
     if (Math.abs(ya) < gen) { Q1.x = a; Q1.y = ya; return Q1; }
     m = (1*a+1*b)/2;
     ym0 = fun(u,m);
     if (Number.isNaN(ym0)) { erg = -999; return erg; }
     ym = differ2(u,m);
     prod = ya* ym;
     if (prod > 0) { a = m; }
     else { b = m; }
   } while (Math.abs(ym) > gen);
     Q1.x = m;
     Q1.y = ym0;
     return Q1;
}
```

```
function integbog(u,a,b) {
   // Bogenlänge einer Kurve von a bis b (mit u als f(x))
   // Bogenlänge = erg
   var z,x,y,d,sum;
   var x0,y0,p,rest;
   var n,i,erg;
   n = 1000;
   d = (b-a)/n;
   z = differ(u,a);
   sum = 1*Math.sqrt(1+z*z);
   if (Number.isNaN(sum)) { erg = -999; return erg; }
   z = differ(u,b);
   y = 1*Math.sqrt(1+z*z);
   if (Number.isNaN(y)) { erg = -999; return erg; }
   sum = sum + y;
   for (i=1; i <= (n-1); i++) {
     x = a + d*i;
     x0 = x-d;
     z = differ(u,x);
     y = 1*Math.sqrt(1+z*z);
     z = differ(u,x0);
     y0 = 1*Math.sqrt(1+z*z);
     p = y0 * y;
     if (p < 0) { erg = 0; return erg; }
     rest = i - Math.floor(i/2)*2;
     if (rest == 1) { sum = sum + 4*y }
     else { sum = sum + 2*y }
   }
   erg = sum*d/3;
   return erg;
}

function integbogX(u,a,b) {
   // x-Wert des Bogenschwerpunktes (S) einer Kurve von a bis b (mit u als f(x))
   // xS = erg / Bogenlänge
   var z,x,y,d,sum;
   var x0,y0,p,rest;
   var n,i,erg;
   n = 1000;
   d = (b-a)/n;
   z = differ(u,a);
   sum = a*Math.sqrt(1+z*z);
   if (Number.isNaN(sum)) { erg = -999; return erg; }
   z = differ(u,b);
   y = b*Math.sqrt(1+z*z);
   if (Number.isNaN(y)) { erg = -999; return erg; }
   sum = sum + y;
   for (i=1; i <= (n-1); i++) {
     x = a + d*i;
     x0 = x-d;
     z = differ(u,x);
     y = x*Math.sqrt(1+z*z);
     z = differ(u,x0);
     y0 = x0*Math.sqrt(1+z*z);
     p = y0 * y;
     if (p < 0) { erg = 0; return erg; }
     rest = i - Math.floor(i/2)*2;
     if (rest == 1) { sum = sum + 4*y }
     else { sum = sum + 2*y }
   }
   erg = sum*d/3;
   return erg;
}

function integbogY(u,a,b) {
   // y-Wert des Bogenschwerpunktes (S) einer Kurve von a bis b (mit u als f(x))
   // yS = erg / Bogenlänge
   var z,z0,x,y,d,sum;
   var x0,y0,p,rest;
   var n,i,erg;
   n = 1000;
   d = (b-a)/n;
   z0 = fun(u,a);
   z = differ(u,a);
   sum = z0 * Math.sqrt(1+z*z);
   if (Number.isNaN(sum)) { erg = -999; return erg; }
   z0 = fun(u,b);
   z = differ(u,b);
   y = z0 * Math.sqrt(1+z*z);
   if (Number.isNaN(y)) { erg = -999; return erg; }
   sum = sum + y;
```

```
   for (i=1; i <= (n-1); i++) {
     x = a + d*i;
     x0 = x-d;
     z0 = fun(u,x);
     z = differ(u,x);
     y = z0 * Math.sqrt(1+z*z);
     z0 = fun(u,x0);
     z = differ(u,x0);
     y0 = z0 * Math.sqrt(1+z*z);
     p = y0 * y;
     if (p < 0) { erg = 0; return erg; }
     rest = i - Math.floor(i/2)*2;
     if (rest == 1) { sum = sum + 4*y }
     else { sum = sum + 2*y }
   }
   erg = sum*d/3;
   return erg;
}

function integflaX(u,a,b) {
   // x-Wert des Flächenschwerpunktes (S) einer Kurve von a bis b (mit u als f(x))
   // xS = erg / Fläche
   var v,x,y,d,sum;
   var x0,y0,p,rest;
   var n,i,erg;
   n = 1000;
   d = (b-a)/n;
   sum = a * fun(u,a);
   if (Number.isNaN(sum)) { erg = -999; return erg; }
   y = b * fun(u,b);
   if (Number.isNaN(y)) { erg = -999; return erg; }
   sum = sum + y;
   for (i=1; i <= (n-1); i++) {
     x = a + d*i;
     x0 = x-d;
     y = x * fun(u,x);
     y0 = x0 * fun(u,x0);
     p = y0 * y;
     if (p < 0) { erg = 0; return erg; }
     rest = i - Math.floor(i/2)*2;
     if (rest == 1) { sum = sum + 4*y }
     else { sum = sum + 2*y }
   }
   erg = sum*d/3;
   return erg;
}

function integflaY(u,a,b) {
   // y-Wert des Flächenschwerpunktes (S) einer Kurve von a bis b (mit u als f(x))
   // yS = erg / Fläche
   var v,x,y,d,sum;
   var x0,y0,p,rest;
   var n,i,erg;
   n = 1000;
   d = (b-a)/n;
   v = '(' + u + ')^2';
   sum = 0.5 * fun(v,a);
   if (Number.isNaN(sum)) { erg = -999; return erg; }
   y = 0.5 * fun(v,b);
   if (Number.isNaN(y)) { erg = -999; return erg; }
   sum = sum + y;
   for (i=1; i <= (n-1); i++) {
     x = a + d*i;
     x0 = x-d;
     y = 0.5 * fun(v,x);
     y0 = 0.5 * fun(v,x0);
     p = y0 * y;
     if (p < 0) { erg = 0; return erg; }
     rest = i - Math.floor(i/2)*2;
     if (rest == 1) { sum = sum + 4*y }
     else { sum = sum + 2*y }
   }
   erg = sum*d/3;
   return erg;
}
```

```
function integfla(u,a,b) {
   // Oberfläche eines Drehkörpers von a bis b (mit u als f(x))
   // Oberfläche = erg
   var z,z0,x,y,d,sum;
   var x0,y0,p,rest;
   var n,i,erg;
   n = 1000;
   d = (b-a)/n;
   z0 = fun(u,a);
   z = differ(u,a);
   sum = z0 * Math.sqrt(1+z*z);
   if (Number.isNaN(sum)) { erg = -999; return erg; }
   z0 = fun(u,b);
   z = differ(u,b);
   y = z0 * Math.sqrt(1+z*z);
   if (Number.isNaN(y)) { erg = -999; return erg; }
   sum = sum + y;
   for (i=1; i <= (n-1); i++) {
     x = a + d*i;
     x0 = x-d;
     z0 = fun(u,x);
     z = differ(u,x);
     y = z0 * Math.sqrt(1+z*z);
     z0 = fun(u,x0);
     z = differ(u,x0);
     y0 = z0 * Math.sqrt(1+z*z);
     p = y0 * y;
     if (p < 0) { erg = 0; return erg; }
     rest = i - Math.floor(i/2)*2;
     if (rest == 1) { sum = sum + 4*y }
     else { sum = sum + 2*y }
   }
   erg = 2*Math.PI*sum*d/3;
   return erg;
}

function integvol(u,a,b) {
   // Volumen eines Drehkörpers von a bis b (mit u als f(x))
   // Volumen = erg
   var v,x,y,d,sum;
   var x0,y0,p,rest;
   var n,i,erg;
   n = 1000;
   d = (b-a)/n;
   v = '(' + u + ')^2';
   sum = Math.PI * fun(v,a);
   if (Number.isNaN(sum)) { erg = -999; return erg; }
   y = Math.PI * fun(v,b);
   if (Number.isNaN(y)) { erg = -999; return erg; }
   sum = sum + y;
   for (i=1; i <= (n-1); i++) {
     x = a + d*i;
     x0 = x-d;
     y = Math.PI * fun(v,x);
     y0 = Math.PI * fun(v,x0);
     p = y0 * y;
     if (p < 0) { erg = 0; return erg; }
     rest = i - Math.floor(i/2)*2;
     if (rest == 1) { sum = sum + 4*y }
     else { sum = sum + 2*y }
   }
   erg = sum*d/3;
   return erg;
}

function integvolX(u,a,b) {
   // x-Wert des Drehkörperschwerpunkt(S) einer Kurve von a bis b (mit u als f(x))
   // xS = erg / Volumen
   var v,x,y,z,d,sum;
   var x0,y0,p,rest;
   var n,i,erg;
   n = 1000;
   d = (b-a)/n;
   z = fun(u,a);
   sum = a * z * z;
   if (Number.isNaN(sum)) { erg = -999; return erg; }
   z = fun(u,b);
   y = b * z * z;
   if (Number.isNaN(y)) { erg = -999; return erg; }
   sum = sum + y;
```

```
   for (i=1; i <= (n-1); i++) {
     x = a + d*i;
     x0 = x-d;
     z = fun(u,x);
     y = x * z * z;
     z = fun(u,x0);
     y0 = x0 * z * z;
     p = y0 * y;
     if (p < 0) { erg = 0; return erg; }
     rest = i - Math.floor(i/2)*2;
     if (rest == 1) { sum = sum + 4*y }
     else { sum = sum + 2*y }
   }
   erg = Math.PI*sum*d/3;
   return erg;
}

function GER(P9,Q9) {
   // Gerade durch die Punkte P9 und Q9, Ergebnisse in L1xy
   // Senkrechte Gerade bei "-999"
   // wenn k = ~, dann L1xy[0] = 1, L1xy[1] = x, L1xy[2] = -999
   // wenn k <> ~, dann L1xy[0] = 0, L1xy[1] = k, L1xy[2] = d
   var P = new TPoint;
   var Q = new TPoint;
   var k1, k2;
   P = P9;
   Q = Q9;
   if (Q.x == P.x) {
      L1xy[0] = 1;
      L1xy[1] = P.x;
      L1xy[2] = -999;
   }
   else {
      k1 = (Q.y - P.y) / (Q.x - P.x);
      k2 = P.y - k1 * P.x;
      L1xy[0] = 0;
      L1xy[1] = k1;
      L1xy[2] = k2;
   }
}

function SGG(A9,B9,C9,D9) {
   // Schnittpunkt der Geraden g(A9,B9) und h(C9,D9)
   // wenn kg = ~, dann L1xy[0] = 1, L1xy[1] = x, L1xy[2] = -999
   // wenn kg <> ~, dann L1xy[0] = 0, L1xy[1] = k, L1xy[2] = d
   // wenn kh = ~, dann L2xy[0] = 1, L2xy[1] = x, L2xy[2] = -999
   // wenn kh <> ~, dann L2xy[0] = 0, L2xy[1] = k, L2xy[2] = d
   // globale Zusatzinformationen in info1, info2 und info3
   var i9,k9,d9;
   var kg,dg,kh,dh;
   var x1,y1;

   GER(A9,B9);
   i9 = L1xy[0];
   k9 = L1xy[1];
   d9 = L1xy[2];

   GER(C9,D9);
   L2xy[0] = L1xy[0];
   L2xy[1] = L1xy[1];
   L2xy[2] = L1xy[2];
   L1xy[0] = i9;
   L1xy[1] = k9;
   L1xy[2] = d9;

   kg = round2(L1xy[1]);
   dg = round2(L1xy[2]);
   info1 = ' y = ' + kg + ' * x + ' + dg;
   kh = round2(L2xy[1]);
   dh = round2(L2xy[2]);
   info2 = ' y = ' + kh + ' * x + ' + dh;
   if ((A9.y == B9.y) && (C9.y == D9.y)) {
      L1xy[1] = 0;
      L1xy[1] = -999;
      L1xy[2] = -999;
      info1 = ' x = ' + kg;
      info2 = ' x = ' + kh;
      return;
   }
```

```
   if ((L1xy[0] == 0) && (L2xy[0] == 0)) {
      mat[1][1] = L1xy[1];
      mat[1][2] = -1;
      mat[1][3] = -L1xy[2];
      mat[2][1] = L2xy[1];
      mat[2][2] = -1;
      mat[2][3] = -L2xy[2];
      L1xy[0] = 0; L1xy[1] = 0; L1xy[2] = 0;
      linglei(mat,2,L1xy);
      return;
   }
   if ((L1xy[0] == 0) && (L2xy[0] == 1)) {
      x1 = L2xy[1];
      y1 = x1 * L1xy[1] + L1xy[2];
      L1xy[0] = 0;
      L1xy[1] = x1;
      L1xy[2] = y1;
      info2 = ' x = ' + kh;
      return;
   }
   if ((L1xy[0] == 1) && (L2xy[0] == 0)) {
      x1 = L1xy[1];
      y1 = x1 * L2xy[1] + L2xy[2];
      L1xy[0] = 0;
      L1xy[1] = x1;
      L1xy[2] = y1
      info1 = ' x = ' + kg;
      return;
   }
   if ((L1xy[0] == 1) && (L2xy[0] == 1)) {
      L1xy[1] = 0;
      L1xy[1] = -999;
      L1xy[2] = -999;
      info1 = ' x = ' + kg;
      info2 = ' x = ' + kh;
      return;
   }
}

function SKG(M9,r9,A9,B9) {
   // Schnittpunkte der Geraden g(A9,B9) mit dem Kreis k(M9,r9)
   // Erster Schnittpunkt: (L1xy[1] / L2xy[1])
   // Zweiter Schnittpunkt: (L1xy[2] / L2xy[2])
   // KEINE Schnittpunkte bei L1xy[0] = 3
   // Geradengleichung in "info1"
   var ig,kg,dg;
   var k1,k2,k3,k4,k5,k6,k7,k8,k9,k10;
   info1 = '';

   GER(A9,B9);
   ig = L1xy[0];
   kg = L1xy[1];
   dg = L1xy[2];
   if (ig == 1) { info1 = ' x = ' + round2(kg); }
   else { info1 = ' y = ' + round2(kg) + ' * x + ' + round2(dg); }
   k1 = M9.x;
   k2 = M9.y;
   k3 = Math.abs(r9);
   if (k3 == 0) { alert('Falscher Kreisradius !'); return; }
   if (ig != 1) {
      k4 = kg;
      k5 = dg;
      k6 = 1 + k4*k4;
      k7 = 2*k4*k5 - 2*k1 - 2*k2*k4;
      k8 = k1*k1 + k2*k2 + k5*k5 - 2*k2*k5 - k3*k3;
      quagl(k6,k7,k8);
      if (L1xy[0] == 3) { return; }
      k9 = L1xy[1];
      k10 = k9 * k4 + k5;
      L2xy[1] = k10;
      k9 = L1xy[2];
      k10 = k9 * k4 + k5;
      L2xy[2] = k10;
   }
```

```
   if (ig == 1) {
      k4 = kg;
      k5 = dg;
      k6 = 1;
      k7 = -2*k2;
      k8 = (k4-k1)*(k4-k1) + k1*k1 + k2*k2 - k3*k3;
      quagl(k6,k7,k8);
      if (L1xy[0] == 3) { return; }
      L2xy[1] = L1xy[1];
      L2xy[2] = L1xy[2];
      L1xy[1] = k4;
      L1xy[2] = k4;
   }
}

function SKK(M11,r11,M22,r22) {
  // Schnitt von Kreis und Kreis
  // Erster Schnittpunkt  S1(L1xy[1]/L2xy[1])
  // Zweiter Schnittpunkt S2(L1xy[2]/L2xy[2])
  // Fehlererkennung mit "-999"
   for (var i = 0; i <= 3; i++) {
     L1xy[i] = 0;
     L2xy[i] = 0;
   }
   var res = 0;
   var k1,k2,k3,k4,k5,k6,k7,k8,k9,k10,k11,k12,k13,k14,k15;
   k1 = M11.x;
   k2 = M11.y;
   k3 = r11;
   k4 = M22.x;
   k5 = M22.y;
   k6 = r22;
   if ((k3 <= 0) || (k6 <= 0)) { L1xy[1] = -999; L1xy[2] = -999; return; }
   if ((k1 == k4) && (k2 == k5)) { L1xy[1] = -999; L1xy[2] = -999; return; }
   if (k2 != k5) {
      k7  = (k1-k4) / (k5-k2);
      k8  = (-k1*k1-k2*k2+k3*k3+k4*k4+k5*k5-k6*k6) / ((k5-k2)* 2);
      k9  = 1 + k7*k7;
      k10 = 2*k7*k8 - 2*k1 - 2*k2*k7;
      k11 = k1*k1+k2*k2-k3*k3+k8*k8-2*k8*k2;
      res = quagl(k9,k10,k11);
      k12 = L1xy[1];
      k13 = k7*k12 + k8;
      k14 = L1xy[2];
      k15 = k7*k14 + k8;
      L2xy[1] = k13;
      L2xy[2] = k15;
   }
   if (k2 == k5) {
      k7  = (k5-k2) / (k1-k4);
      k8  = (k1*k1+k2*k2-k3*k3-k4*k4-k5*k5+k6*k6) / ((k1-k4) * 2);
      k9  = 1 + k7*k7;
      k10 = 2*k7*k8 - 2*k2 - 2*k1*k7;
      k11 = k1*k1+k2*k2-k3*k3+k8*k8-2*k8*k1;
      res = quagl(k9,k10,k11);
      k13 = L1xy[1];
      L2xy[1] = k13;
      k12 = k7*k13 + k8;
      L1xy[1] = k12;
      k15 = L1xy[2];
      L2xy[2] = k15;
      k14 = k7*k15 + k8;
      L1xy[2] = k14;
   }
   if (res == 3) { L1xy[1] = -999; L1xy[2] = -999; }
}

function SSM(A,B) {
   // Streckensymmetrale y = k*x+d von AB
   // mit k = L1xy[1] und d = L1xy[2]
   // und Streckenmittelpunkt L2xy[1] = M9.x und L2xy[2] = M9.y
   // Senkrechte Gerade bei "-999"
   var M9 = new TPoint;
   var k1,k2;
   L1xy[0] = 0;
   L2xy[0] = 0;
   M9.x = (A.x + B.x) / 2;
   M9.y = (A.y + B.y) / 2;
```

```
   if (A.y == B.y) {
      L2xy[1] = M9.x;
      L2xy[2] = M9.y;
      L1xy[1] = M9.x;
      L1xy[2] = -999;
   }
   else {
     k1 = (A.x-B.x) / (B.y-A.y);
     k2 = M9.y - k1 * M9.x;
     L1xy[1] = k1;
     L1xy[2] = k2;
     if (B.x == A.x) {
         L2xy[1] = A.x;
         L2xy[2] = M9.y;
     }
     else {
         k3 = (B.y-A.y) / (B.x-A.x);
         k4 = A.y - k3 * A.x;
         L2xy[1] = (-1) * (k4-k2) / (k3-k1);
         L2xy[2] = k3*L2xy[1] + k4;
     }
   }
}

function WSM(P,Q,R) {
   // Winkelsymmetrale y = k*x + d von w(PQR)
   // L1xy[1] = k und L1xy[2] = d
   // Senkrechte Gerade bei "-999"
   var k1,k2,k3,k4,k5,k6;
   L1xy[0] = 0;
   L2xy[0] = 0;
   k3 = Math.sqrt((P.x-Q.x)*(P.x-Q.x) + (P.y-Q.y)*(P.y-Q.y));
   k4 = Math.sqrt((R.x-Q.x)*(R.x-Q.x) + (R.y-Q.y)*(R.y-Q.y));
   k5 = (P.x-Q.x)/k3+(R.x-Q.x)/k4;
   k6 = (P.y-Q.y)/k3+(R.y-Q.y)/k4;
   if (k5 == 0) {
     L1xy[1] = Q.x;
     L1xy[2] = -999;
   }
   else {
     k1 = k6 / k5;
     k2 = Q.y - k1 * Q.x;
     L1xy[1] = k1;
     L1xy[2] = k2;
   }
}

function LOT(C,A,B) {
   // Das Lot y = k*x + d von C auf AB
   // mit k = L1xy[1] und d = L1xy[2]
   // Lotfußpunkt C mit C.x = L2xy[1] und C.y = L2xy[2]
   // Senkrechte Gerade bei "-999"
   var k1,k2,k3,k4;
   L1xy[0] = 0;
   L2xy[0] = 0;
   if (A.y == B.y) {
      L2xy[1] = C.x;
      L2xy[2] = A.y;
      L1xy[1] = C.x;
      L1xy[2] = -999;
   }
   else {
     k1 = (A.x-B.x) / (B.y-A.y);
     k2 = C.y - k1 * C.x;
     L1xy[1] = k1;
     L1xy[2] = k2;
     if (B.x == A.x) {
         L2xy[1] = A.x;
         L2xy[2] = C.y;
     }
     else {
         k3 = (B.y-A.y) / (B.x-A.x);
         k4 = A.y - k3 * A.x;
         L2xy[1] = (-1)*(k4-k2) / (k3-k1);
         L2xy[2] = k3*L2xy[1] + k4;
     }
   }
}
```

```
function deta(b,n) {
// berechnung der determinante
// b bzw. a = koeffizienten-matrix "mat[i][j]", n = anzahl
  var i,j,k,m,c;
  var r,s,error;
  error = 0; r = 0; c = 1;
  // aufsuchen des grössten anfangs-koeffizienten
  var a = new Array(n);
  for (var j = 0; j <= n; j++) { a[j] = new Array(m); }
  for (var i = 1; i <= n; i++) {
    for (var j = 0; j <= n; j++) { a[i][j] = b[i][j]; }
  }
  for (var i = 1; i <= n-1; i++) {
      m = i;
      for (var k = i+1; k <= n; k++) {
          if (Math.abs(a[k][i]) > Math.abs(a[m][i])) { m = k; }
      }
  // die i-te zeile mit der m-ten zeile tauschen
      if (i != m) {
         c = -c;
         for (var j = i; j <= n; j++) {
             s = a[i][j];
             a[i][j] = a[m][j];
             a[m][j] = s;
         }
      }
  // elimination von x[i] aus der k-ten gleichung
      for (var k = i+1; k <= n; k++) {
         if (a[i][i] == 0) { error = -999; return error; }
         s = a[k][i] / a[i][i];
         for (var j = i+1; j <= n; j++) {
         a[k][j]= a[k][j]-a[i][j]*s;
         }
      }
   }
  // eigentliche berechnung der determinante
   r = 1;
   for (var i = n; i >= 1; i = i-1) { r = r * a[i][i]; }
   r = c * r;
   return r;
}

function linglei(b,n,z) {
// lösung eines linearen gleichungsystems in n variablen
// b bzw. a = koeffizienten-matrix "mat[i][j]"
// z bzw. x = lösungs-vektor "L1xy[i]"
  for (var i = 0; i <= 3; i++) { L1xy[i] = 0; }
  var i,j,k,m;
  var s;
  var error = 0;
  var a = new Array(n);
  for (var j = 0; j <= n+1; j++) { a[j] = new Array(n+1); }
  for (var i = 1; i <= n; i++) {
    for (var j = 0; j <= n+1; j++) { a[i][j] = b[i][j]; }
  }
  var x = new Array(n);
  x[0] = 0;
  for (var i = 0; i <= n; i++) { x[i] = z[i]; }
  // aufsuchen des grössten anfangs-koeffizienten
  for (var i = 1; i <= n-1; i++) {
      m = i;
      for (var k = i+1; k <= n; k++) {
          if (Math.abs(a[k][i]) > Math.abs(a[m][i])) { m = k; }

          // die i-te zeile mit der m-ten zeile tauschen
          if (i != m) {
             for (var j = i; j <= n+1; j++) {
                 s = a[i][j];
                 a[i][j] = a[m][j];
                 a[m][j] = s;
             }
          }
      }
  // elimination von x[i] aus der k-ten gleichung
      for (var k = i+1; k <= n; k++) {
         if (a[i][i] == 0) { error = -999; x[0] = error; return error; }
         s = a[k][i] / a[i][i];
         for (var j = i+1; j <= n+1; j++) {
         a[k][j]= a[k][j]-a[i][j]*s;
         }
      }
   }
```

```
  // eigentliche berechnung der lösungen
   for (var i = n; i >= 1; i = i-1) {
   s = a[i][n+1];
   for (var j = i+1; j <= n; j++) { s = s - a[i][j]*x[j]; }
   if (a[i][i] == 0) { error = -999; x[0] = error; return error; }
   x[i]= s / a[i][i];
   }
   L1xy = x;
}

function quagl(a,b,c) {
// Lösungen einer quadratischen gleichung
// Lösungsart L1xy[0], ( 1 = eine, 2 = zwei, 3 = keine )
// Lösungen L1xy[1] und L1xy[2]
  var disk,x1,x2,erg;
  if (a == 0) {
     erg = 0;
     x1 = -c/b;
     x2 = x1;
     x1 = round2(x1);
     x2 = round2(x2);
     L1xy[0] = erg; L1xy[1] = x1; L1xy[2] = x2;
     info = ' Keine quadratische Gleichung ';
     return erg;
  }
  disk = b*b - 4*a*c;
  if (disk == 0) {
     erg = 1;
     x1 = -b/2/a;
     x2 = x1;
     x1 = round2(x1);
     x2 = round2(x2);
     L1xy[0] = erg; L1xy[1] = x1; L1xy[2] = x2;
     info = ' Eine reelle Doppel-Lösung ';
     return erg;
  }
  if (disk > 0) {
     erg = 2;
     x1 = (-b + Math.sqrt(disk))/(2*a);
     x2 = (-b - Math.sqrt(disk))/(2*a);
     x1 = round2(x1);
     x2 = round2(x2);
     L1xy[0] = erg; L1xy[1] = x1; L1xy[2] = x2;
     info = ' Zwei reelle Lösungen ';
     return erg;
  }
  if (disk < 0) {
     erg = 3;
     x1 = -b/(2*a);
     x2 = Math.sqrt(-disk)/(2*a);
     x1 = round2(x1);
     x2 = round2(x2);
     z1 = x1 + 'i*' + round2(x2);
     z2 = x1 - 'i*' + round2(x2);
     L1xy[0] = erg; L1xy[1] = x1; L1xy[2] = x2;
     info = ' Zwei Komplexe Lösungen ';
     return erg;
  }
}

// ---------------------------------------------------------------------------
// Routinen zur Vektorrechnung zweidimensional (2D) und dreidimensional (3D)
// ---------------------------------------------------------------------------

function vekmal(i,A) {
   // Multiplikation eines Vektors mit einer Zahl (2D)
   var Z = new TPoint;
   Z.x = i*A.x;
   Z.y = i*A.y;
   return Z;
}

function veksum(A,B) {
   // Addition zweier Vektoren (2D)
   var Z = new TPoint;
   Z.x = 1*(A.x + B.x);
   Z.y = 1*(A.y + B.y);
   return Z;
}
```

```
function veklin(s,t,A,B) {
   // Linearkombination zweier Vektoren (2D)
   var Z = new TPoint;
   Z.x = s*A.x + t*B.x;
   Z.y = s*A.y + t*B.y;
   return Z;
}

function vekmul(A,B) {
   // Skalarprodukt zweier Vektoren (2D)
   var z = A.x * B.x + A.y * B.y;
   return z;
}

function Shap(P,Q) {
   // Streckenhalbierungspunkt (2D)
   var H2 = new TPoint;
   H2.x = (P.x + Q.x) / 2;
   H2.y = (P.y + Q.y) / 2;
   return H2;
}

function Spunkt(P,Q,R) {
   // Schwerpunkt (2D)
   var H2 = new TPoint;
   H2.x = (P.x + Q.x + R.x) / 3;
   H2.y = (P.y + Q.y + R.y) / 3;
   return H2;
}

function Slen(P,Q) {
  // Distanz zweier Punkte (2D)
  var z = (P.x - Q.x)*(P.x - Q.x) + (P.y - Q.y)*(P.y - Q.y);
  return Math.sqrt(z);
}

function Umfang(P,Q,R) {
  // Umfang eines Dreiecks (2D)
  var a1,b1,c1,u1;
  a1 = Slen(P,Q);
  b1 = Slen(Q,R);
  c1 = Slen(R,P);
  u1 = (a1 + b1 + c1);
  return u1;
}

function Flaeche(P,Q,R) {
  // Fläche eines Dreiecks (2D)
  var a1,b1,c1,u1,s1,erg;
  a1 = Slen(P,Q);
  b1 = Slen(Q,R);
  c1 = Slen(R,P);
  u1 = (a1 + b1 + c1) / 2;
  s1 = u1*(u1-a1)*(u1-b1)*(u1-c1);
  if (s1 <= 0) { erg = 0; }
  else { erg = Math.sqrt(s1); }
  return erg;
}

function Winkel(P,Q,R) {
  // Winkel w bei Punkt Q (2D)
  var a1,b1,c1,x1,y1,z1;
  a1 = Slen(Q,R);
  b1 = Slen(P,R);
  c1 = Slen(P,Q);
  x1 = (a1*a1 + c1*c1 - b1*b1) / (2*a1*c1);
  y1 = Math.acos(x1);
  z1 = deg(y1);
  return z1;
}

function vekmal3(n,P3) {
   // Multiplikation eines Vektors mit einer Zahl (3D)
   var H3 = new TPoint3;
   H3.x = n*P3.x;
   H3.y = n*P3.y;
   H3.z = n*P3.z;
   return H3;
}
```

```
function veksum3(P3,Q3) {
   // Addition zweier Vektoren (3D)
   var H3 = new TPoint3;
   H3.x = 1*P3.x + 1*Q3.x;
   H3.y = 1*P3.y + 1*Q3.y;
   H3.z = 1*P3.z + 1*Q3.z;
   return H3;
}

function veklin3(s,t,P3,Q3) {
   // Linearkombination zweier Vektoren (3D)
   var H3 = new TPoint3;
   H3.x = s*P3.x + t*Q3.x;
   H3.y = s*P3.y + t*Q3.y;
   H3.z = s*P3.z + t*Q3.z;
   return H3;
}

function vekmul3(P3,Q3) {
   // Skalarprodukt zweier Vektoren (3D)
   var z = P3.x * Q3.x + P3.y * Q3.y + P3.z * Q3.z;
   return z;
}

function vekkreuz3(P3,Q3) {
   // Kreuzprodukt zweier Vektoren (3D)
   var H3 = new TPoint3;
   H3.x = P3.y*Q3.z - P3.z*Q3.y;
   H3.y = -(P3.x*Q3.z - P3.z*Q3.x);
   H3.z = P3.x*Q3.y - P3.y*Q3.x;
   return H3;
}

function vekspat3(P3,Q3,R3) {
   // Spatprodukt zweier Vektoren (3D)
   var z = (P3.y*Q3.z - P3.z*Q3.y)*R3.x + (P3.z*Q3.x - P3.x*Q3.z)*R3.y +
           (P3.x*Q3.y - P3.y*Q3.x)*R3.z;
   return Math.abs(z);
}

function Shap3(P,Q) {
   // Streckenhalbierungspunkt (3D)
   var H3 = new TPoint3;
   H3.x = (P.x + Q.x + P.z) / 2;
   H3.y = (P.y + Q.y + Q.z) / 2;
   return H3;
}

function Spunkt3(P,Q,R) {
   // Schwerpunkt (3D)
   var H3 = new TPoint3;
   H3.x = (P.x + Q.x + R.x) / 3;
   H3.y = (P.y + Q.y + R.y) / 3;
   H3.z = (P.z + Q.z + R.z) / 3;
   return H3;
}

function Slen3(P3,Q3) {
  // Distanz zweier Punkte (3D)
  var z = (P3.x - Q3.x)*(P3.x - Q3.x) + (P3.y - Q3.y)*(P3.y - Q3.y) +
          (P3.z - Q3.z)*(P3.z - Q3.z);
  return Math.sqrt(z);
}

function Umfang3(P3,Q3,R3) {
  // Umfang eines Dreiecks (3D)
  var a1,b1,c1,u1;
  a1 = Slen3(P3,Q3);
  b1 = Slen3(Q3,R3);
  c1 = Slen3(R3,P3);
  u1 = (a1 + b1 + c1);
  return u1;
}
```

```
function Flaeche3(P3,Q3,R3) {
  // Fläche eines Dreiecks (3D)
  var a1,b1,c1,u1,s1,erg;
  a1 = Slen3(P3,Q3);
  b1 = Slen3(Q3,R3);
  c1 = Slen3(R3,P3);
  u1 = (a1 + b1 + c1) / 2;
  s1 = u1*(u1-a1)*(u1-b1)*(u1-c1);
  if (s1 <= 0) { erg = 0; }
  else { erg = Math.sqrt(s1); }
  return erg;
}

function Winkel3(P3,Q3,R3) {
  // Winkel w bei Punkt Q3 (3D)
  var a1,b1,c1,x1,y1,z1;
  a1 = Slen3(Q3,R3);
  b1 = Slen3(P3,R3);
  c1 = Slen3(P3,Q3);
  x1 = (a1*a1 + c1*c1 - b1*b1) / (2*a1*c1);
  y1 = Math.acos(x1);
  z1 = deg(y1);
  return z1;
}

// ---------------------------------------------------------------------------
// Ende von "mathe.js"
// ---------------------------------------------------------------------------
```

[4.5] Die JavaScript-Datei *„parser.js“*

```
// ---------------------------------------------------------------------------
// "parser.js" based on "ndef.parser" by Raphael Graf,
//  modified by Herbert Paukert, Version 4.0 am 10.12.2018
// ---------------------------------------------------------------------------

var a,b,c,d,e,f,g,h,I,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z;

function fun(s,x)  {
// Ein geparster Funktionterm "s" in nur EINER Variablen x.
   var t,z;
   t = Parser.parse(s);
   z = t.evaluate({ x: x });
   return z;
}

function formel(ss) {
// Eine geparste Formel "ss" in 26 Variablen a,b,...i,j.
   var tt,zz;
   tt = Parser.parse(s);
   zz = tt.evaluate({ a:a,b:b,c:c,d:d,e:e,f:f,g:g,h:h,i:i,j:j,k:k,l:l,m:m,n:n,o:o,p:p,q:q,
                      r:r,s:s,t:t,u:u,v:v,w:w,x:x,y:y,z:z });
   return zz;
}

var Parser = (function (scope) {
  function object(o) {
    function F() {}
    F.prototype = o;
    return new F();
  }

  var TNUMBER = 0;
  var TOP1 = 1;
  var TOP2 = 2;
  var TVAR = 3;
  var TFUNCALL = 4;

  function Token(type_, index_, prio_, number_) {
    this.type_ = type_;
    this.index_ = index_ || 0;
    this.prio_ = prio_ || 0;
    this.number_ = (number_ !== undefined && number_ !== null) ? number_ : 0;
    this.toString = function () {
      switch (this.type_) {
      case TNUMBER:
        return this.number_;
      case TOP1:
      case TOP2:
      case TVAR:
        return this.index_;
      case TFUNCALL:
        return "CALL";
      default:
        return "Invalid Token";
      }
    };
  }

  function Expression(tokens, ops1, ops2, functions) {
    this.tokens = tokens;
    this.ops1 = ops1;
    this.ops2 = ops2;
    this.functions = functions;
  }

  // Based on http://www.json.org/json2.js
    var cx = /[\u0000\u00ad\u0600-\u0604\u070f\u17b4\u17b5\u200c-\u200f\u2028-\u202f\u2060-
           \u206f\ufeff\ufff0-\uffff]/g,
        escapable = /[\\\'\x00-\x1f\x7f-\x9f\u00ad\u0600-\u0604\u070f\u17b4\u17b5\u200c-
                  \u200f\u2028-\u202f\u2060-\u206f\ufeff\ufff0-\uffff]/g,
        meta = { // table of character substitutions
            '\b': '\\b',
            '\t': '\\t',
            '\n': '\\n',
            '\f': '\\f',
            '\r': '\\r',
            "'" : "\\'",
            '\\': '\\\\'
        };
```

```
function escapeValue(v) {
  if (typeof v === "string") {
    escapable.lastIndex = 0;
        return escapable.test(v) ?
            "'" + v.replace(escapable, function (a) {
                var c = meta[a];
                return typeof c === 'string' ? c :
                    '\\u' + ('0000' + a.charCodeAt(0).toString(16)).slice(-4);
            }) + "'" :
            "'" + v + "'";
  }
  return v;
}

Expression.prototype = {
  simplify: function (values) {
    values = values || {};
    var nstack = [];
    var newexpression = [];
    var n1;
    var n2;
    var f;
    var L = this.tokens.length;
    var item;
    var i = 0;
    for (i = 0; i < L; i++) {
      item = this.tokens[i];
      var type_ = item.type_;
      if (type_ === TNUMBER) {
        nstack.push(item);
      }
      else if (type_ === TVAR && (item.index_ in values)) {
        item = new Token(TNUMBER, 0, 0, values[item.index_]);
        nstack.push(item);
      }
      else if (type_ === TOP2 && nstack.length > 1) {
        n2 = nstack.pop();
        n1 = nstack.pop();
        f = this.ops2[item.index_];
        item = new Token(TNUMBER, 0, 0, f(n1.number_, n2.number_));
        nstack.push(item);
      }
      else if (type_ === TOP1 && nstack.length > 0) {
        n1 = nstack.pop();
        f = this.ops1[item.index_];
        item = new Token(TNUMBER, 0, 0, f(n1.number_));
        nstack.push(item);
      }
      else {
        while (nstack.length > 0) {
          newexpression.push(nstack.shift());
        }
        newexpression.push(item);
      }
    }
    while (nstack.length > 0) {
      newexpression.push(nstack.shift());
    }
    return new Expression(newexpression, object(this.ops1), object(this.ops2),
                          object(this.functions));
  },

  substitute: function (variable, expr) {
    if (!(expr instanceof Expression)) {
      expr = new Parser().parse(String(expr));
    }
    var newexpression = [];
    var L = this.tokens.length;
    var item;
    var i = 0;
    for (i = 0; i < L; i++) {
      item = this.tokens[i];
      var type_ = item.type_;
      if (type_ === TVAR && item.index_ === variable) {
        for (var j = 0; j < expr.tokens.length; j++) {
          var expritem = expr.tokens[j];
          var replitem = new Token(expritem.type_, expritem.index_, expritem.prio_,
                                   expritem.number_);
          newexpression.push(replitem);
        }
      }
```

```
      else {
        newexpression.push(item);
      }
    }

    var ret = new Expression(newexpression, object(this.ops1), object(this.ops2),
                             object(this.functions));
    return ret;
  },

  evaluate: function (values) {
    values = values || {};
    var nstack = [];
    var n1;
    var n2;
    var f;
    var L = this.tokens.length;
    var item;
    var i = 0;
    for (i = 0; i < L; i++) {
      item = this.tokens[i];
      var type_ = item.type_;
      if (type_ === TNUMBER) {
        nstack.push(item.number_);
      }
      else if (type_ === TOP2) {
        n2 = nstack.pop();
        n1 = nstack.pop();
        f = this.ops2[item.index_];
        nstack.push(f(n1, n2));
      }
      else if (type_ === TVAR) {
        if (item.index_ in values) {
          nstack.push(values[item.index_]);
        }
        else if (item.index_ in this.functions) {
          nstack.push(this.functions[item.index_]);
        }
        else {
          throw new Error("undefined variable: " + item.index_);
        }
      }
      else if (type_ === TOP1) {
        n1 = nstack.pop();
        f = this.ops1[item.index_];
        nstack.push(f(n1));
      }
      else if (type_ === TFUNCALL) {
        n1 = nstack.pop();
        f = nstack.pop();
        if (f.apply && f.call) {
          if (Object.prototype.toString.call(n1) == "[object Array]") {
            nstack.push(f.apply(undefined, n1));
          }
          else {
            nstack.push(f.call(undefined, n1));
          }
        }
        else {
          throw new Error(f + " is not a function");
        }
      }
      else {
        throw new Error("invalid Expression");
      }
    }
    if (nstack.length > 1) {
      throw new Error("invalid Expression (parity)");
    }
    return nstack[0];
  },

  toString: function (toJS) {
    var nstack = [];
    var n1;
    var n2;
    var f;
    var L = this.tokens.length;
    var item;
    var i = 0;
```

```
    for (i = 0; i < L; i++) {
      item = this.tokens[i];
      var type_ = item.type_;
      if (type_ === TNUMBER) {
        nstack.push(escapeValue(item.number_));
      }
      else if (type_ === TOP2) {
        n2 = nstack.pop();
        n1 = nstack.pop();
        f = item.index_;
        if (toJS && f == "^") {
          nstack.push("Math.pow(" + n1 + "," + n2 + ")");
        }
        else {
          nstack.push("(" + n1 + f + n2 + ")");
        }
      }
      else if (type_ === TVAR) {
        nstack.push(item.index_);
      }
      else if (type_ === TOP1) {
        n1 = nstack.pop();
        f = item.index_;
        if (f === "-") {
          nstack.push("(" + f + n1 + ")");
        }
        else {
          nstack.push(f + "(" + n1 + ")");
        }
      }
      else if (type_ === TFUNCALL) {
        n1 = nstack.pop();
        f = nstack.pop();
        nstack.push(f + "(" + n1 + ")");
      }
      else {
        throw new Error("invalid Expression");
      }
    }
    if (nstack.length > 1) {
      throw new Error("invalid Expression (parity)");
    }
    return nstack[0];
  },

  variables: function () {
    var L = this.tokens.length;
    var vars = [];
    for (var i = 0; i < L; i++) {
      var item = this.tokens[i];
      if (item.type_ === TVAR && (vars.indexOf(item.index_) == -1)) {
        vars.push(item.index_);
      }
    }
    return vars;
  },
  toJSFunction: function (param, variables) {
    var f = new Function(param, "with(Parser.values) { return " +
                         this.simplify(variables).toString(true) + "; }");
    return f;
  }
};

function add(a, b) {
  return Number(a) + Number(b);
}
function sub(a, b) {
  return a - b;
}
function mul(a, b) {
  return a * b;
}
function div(a, b) {
  return a / b;
}
function mod(a, b) {
  return a % b;
}
function concat(a, b) {
  return "" + a + b;
}
```

```
function neg(a) { return -a; }

function random(a) { return Math.random() * (a || 1); }

function fac(a) { //a!
  a = Math.floor(a);
  var b = a;
  while (a > 1) {
    b = b * (--a);
  }
  return b;
}

function rd2(a) { return Math.round(100*a) / 100; }

function rd4(a) { return Math.round(10000*a) / 10000; }

function lg(a) { return Math.log(a) / Math.log(10); }

function deg(a) { return a * 180 / Math.PI; }

function rad(a) { return a * Math.PI / 180; }

// TODO: use hypot that doesn't overflow
function pyt(a, b) {
  return Math.sqrt(a * a + b * b);
}

function append(a, b) {
  if (Object.prototype.toString.call(a) != "[object Array]") {
    return [a, b];
  }
  a = a.slice();
  a.push(b);
  return a;
}

function Parser() {
  this.success = false;
  this.errormsg = "";
  this.expression = "";

  this.pos = 0;

  this.tokennumber = 0;
  this.tokenprio = 0;
  this.tokenindex = 0;
  this.tmpprio = 0;

  this.ops1 = {
    "sin": Math.sin,
    "cos": Math.cos,
    "tan": Math.tan,
    "asin": Math.asin,
    "acos": Math.acos,
    "atan": Math.atan,
    "sqrt": Math.sqrt,
    "log": Math.log,
    "abs": Math.abs,
    "ceil": Math.ceil,
    "floor": Math.floor,
    "round": Math.round,
    "-": neg,
    "exp": Math.exp
  };

  this.ops2 = {
    "+": add,
    "-": sub,
    "*": mul,
    "/": div,
    "%": mod,
    "^": Math.pow,
    ",": append,
    "||": concat
  };
```

```
  this.functions = {
    "random": random,
    "fac": fac,
    "min": Math.min,
    "max": Math.max,
    "pyt": pyt,
    "pow": Math.pow,
    "atan2": Math.atan2,
    "rd2": rd2,
    "rd4": rd4,
    "lg": lg,
    "deg": deg,
    "rad": rad
  };

  this.consts = {
    "E": Math.E,
    "PI": Math.PI
  };
}

Parser.parse = function (expr) {
  return new Parser().parse(expr);
};

Parser.evaluate = function (expr, variables) {
  return Parser.parse(expr).evaluate(variables);
};

Parser.Expression = Expression;

Parser.values = {
  sin: Math.sin,
  cos: Math.cos,
  tan: Math.tan,
  asin: Math.asin,
  acos: Math.acos,
  atan: Math.atan,
  sqrt: Math.sqrt,
  log: Math.log,
  abs: Math.abs,
  ceil: Math.ceil,
  floor: Math.floor,
  round: Math.round,
  random: random,
  fac: fac,
  exp: Math.exp,
  min: Math.min,
  max: Math.max,
  pyt: pyt,
  pow: Math.pow,
  atan2: Math.atan2,
  rd2: rd2,
  rd4: rd4,
  lg: lg,
  deg: deg,
  rad: rad,
  E: Math.E,
  PI: Math.PI
};

var PRIMARY  = 1 << 0;
var OPERATOR = 1 << 1;
var FUNCTION = 1 << 2;
var LPAREN   = 1 << 3;
var RPAREN   = 1 << 4;
var COMMA    = 1 << 5;
var SIGN     = 1 << 6;
var CALL     = 1 << 7;

Parser.prototype = {
  parse: function (expr) {
    this.errormsg = "";
    this.success = true;
    var operstack = [];
    var tokenstack = [];
    this.tmpprio = 0;
    var expected = (PRIMARY | LPAREN | FUNCTION | SIGN);
    var noperators = 0;
    this.expression = expr;
    this.pos = 0;
```

```
while (this.pos < this.expression.length) {
  if (this.isOperator()) {
    if (this.isSign() && (expected & SIGN)) {
      if (this.isNegativeSign()) {
        this.tokenprio = 2;
        this.tokenindex = "-";
        noperators++;
        this.addfunc(tokenstack, operstack, TOP1);
      }
      expected = (PRIMARY | LPAREN | FUNCTION | SIGN);
    }
    else if (this.isComment()) {

    }
    else {
      if ((expected & OPERATOR) === 0) {
        this.error_parsing(this.pos, "unexpected operator");
      }
      noperators += 2;
      this.addfunc(tokenstack, operstack, TOP2);
      expected = (PRIMARY | LPAREN | FUNCTION | SIGN);
    }
  }
  else if (this.isNumber()) {
    if ((expected & PRIMARY) === 0) {
      this.error_parsing(this.pos, "unexpected number");
    }
    var token = new Token(TNUMBER, 0, 0, this.tokennumber);
    tokenstack.push(token);

    expected = (OPERATOR | RPAREN | COMMA);
  }
  else if (this.isString()) {
    if ((expected & PRIMARY) === 0) {
      this.error_parsing(this.pos, "unexpected string");
    }
    var token = new Token(TNUMBER, 0, 0, this.tokennumber);
    tokenstack.push(token);

    expected = (OPERATOR | RPAREN | COMMA);
  }
  else if (this.isLeftParenth()) {
    if ((expected & LPAREN) === 0) {
      this.error_parsing(this.pos, "unexpected \"(\"");
    }

    if (expected & CALL) {
      noperators += 2;
      this.tokenprio = -2;
      this.tokenindex = -1;
      this.addfunc(tokenstack, operstack, TFUNCALL);
    }

    expected = (PRIMARY | LPAREN | FUNCTION | SIGN);
  }
  else if (this.isRightParenth()) {
    if ((expected & RPAREN) === 0) {
      this.error_parsing(this.pos, "unexpected \")\"");
    }

    expected = (OPERATOR | RPAREN | COMMA | LPAREN | CALL);
  }
  else if (this.isComma()) {
    if ((expected & COMMA) === 0) {
      this.error_parsing(this.pos, "unexpected \",\"");
    }
    this.addfunc(tokenstack, operstack, TOP2);
    noperators += 2;
    expected = (PRIMARY | LPAREN | FUNCTION | SIGN);
  }
  else if (this.isConst()) {
    if ((expected & PRIMARY) === 0) {
      this.error_parsing(this.pos, "unexpected constant");
    }
    var consttoken = new Token(TNUMBER, 0, 0, this.tokennumber);
    tokenstack.push(consttoken);
    expected = (OPERATOR | RPAREN | COMMA);
  }
```

```
      else if (this.isOp2()) {
        if ((expected & FUNCTION) === 0) {
          this.error_parsing(this.pos, "unexpected function");
        }
        this.addfunc(tokenstack, operstack, TOP2);
        noperators += 2;
        expected = (LPAREN);
      }
      else if (this.isOp1()) {
        if ((expected & FUNCTION) === 0) {
          this.error_parsing(this.pos, "unexpected function");
        }
        this.addfunc(tokenstack, operstack, TOP1);
        noperators++;
        expected = (LPAREN);
      }
      else if (this.isVar()) {
        if ((expected & PRIMARY) === 0) {
          this.error_parsing(this.pos, "unexpected variable");
        }
        var vartoken = new Token(TVAR, this.tokenindex, 0, 0);
        tokenstack.push(vartoken);

        expected = (OPERATOR | RPAREN | COMMA | LPAREN | CALL);
      }
      else if (this.isWhite()) {
      }
      else {
        if (this.errormsg === "") {
          this.error_parsing(this.pos, "unknown character");
        }
        else {
          this.error_parsing(this.pos, this.errormsg);
        }
      }
    }

    if (this.tmpprio < 0 || this.tmpprio >= 10) {
      this.error_parsing(this.pos, "unmatched \"()\"");
    }

    while (operstack.length > 0) {
      var tmp = operstack.pop();
      tokenstack.push(tmp);
    }

    if (noperators + 1 !== tokenstack.length) {
      //print(noperators + 1);
      //print(tokenstack);
      this.error_parsing(this.pos, "parity");
    }

    return new Expression(tokenstack, object(this.ops1), object(this.ops2),
                          object(this.functions));
  },

  evaluate: function (expr, variables) {
    return this.parse(expr).evaluate(variables);
  },

  error_parsing: function (column, msg) {
    this.success = false;
    this.errormsg = "parse error [column " + (column) + "]: " + msg;
    throw new Error(this.errormsg);
  },

  addfunc: function (tokenstack, operstack, type_) {
    var operator = new Token(type_, this.tokenindex, this.tokenprio + this.tmpprio, 0);
    while (operstack.length > 0) {
      if (operator.prio_ <= operstack[operstack.length - 1].prio_) {
        tokenstack.push(operstack.pop());
      }
      else {
        break;
      }
    }
    operstack.push(operator);
  },
```

```
isNumber: function () {
  var r = false;
  var str = "";
  while (this.pos < this.expression.length) {
    var code = this.expression.charCodeAt(this.pos);
    if ((code >= 48 && code <= 57) || code === 46) {
      str += this.expression.charAt(this.pos);
      this.pos++;
      this.tokennumber = parseFloat(str);
      r = true;
    }
    else {
      break;
    }
  }
  return r;
},

// Ported from the yajjl JSON parser at http://code.google.com/p/yajjl/
unescape: function(v, pos) {
  var buffer = [];
  var escaping = false;

  for (var i = 0; i < v.length; i++) {
    var c = v.charAt(i);

    if (escaping) {
      switch (c) {
      case "'":
        buffer.push("'");
        break;
      case '\\':
        buffer.push('\\');
        break;
      case '/':
        buffer.push('/');
        break;
      case 'b':
        buffer.push('\b');
        break;
      case 'f':
        buffer.push('\f');
        break;
      case 'n':
        buffer.push('\n');
        break;
      case 'r':
        buffer.push('\r');
        break;
      case 't':
        buffer.push('\t');
        break;
      case 'u':
        // interpret the following 4 characters as the hex of the unicode code point
        var codePoint = parseInt(v.substring(i + 1, i + 5), 16);
        buffer.push(String.fromCharCode(codePoint));
        i += 4;
        break;
      default:
        throw this.error_parsing(pos + i, "Illegal escape sequence: '\\" + c + "'");
      }
      escaping = false;
    } else {
      if (c == '\\') {
        escaping = true;
      } else {
        buffer.push(c);
      }
    }
  }

  return buffer.join('');
},
```

```
isString: function () {
  var r = false;
  var str = "";
  var startpos = this.pos;
  if (this.pos < this.expression.length && this.expression.charAt(this.pos) == "'") {
    this.pos++;
    while (this.pos < this.expression.length) {
      var code = this.expression.charAt(this.pos);
      if (code != "'" || str.slice(-1) == "\\") {
        str += this.expression.charAt(this.pos);
        this.pos++;
      }
      else {
        this.pos++;
        this.tokennumber = this.unescape(str, startpos);
        r = true;
        break;
      }
    }
  }
  return r;
},

isConst: function () {
  var str;
  for (var i in this.consts) {
    if (true) {
      var L = i.length;
      str = this.expression.substr(this.pos, L);
      if (i === str) {
        this.tokennumber = this.consts[i];
        this.pos += L;
        return true;
      }
    }
  }
  return false;
},

isOperator: function () {
  var code = this.expression.charCodeAt(this.pos);
  if (code === 43) { // +
    this.tokenprio = 0;
    this.tokenindex = "+";
  }
  else if (code === 45) { // -
    this.tokenprio = 0;
    this.tokenindex = "-";
  }
  else if (code === 124) { // |
    if (this.expression.charCodeAt(this.pos + 1) === 124) {
      this.pos++;
      this.tokenprio = 0;
      this.tokenindex = "||";
    }
    else {
      return false;
    }
  }
  else if (code === 42) { // *
    this.tokenprio = 1;
    this.tokenindex = "*";
  }
  else if (code === 47) { // /
    this.tokenprio = 2;
    this.tokenindex = "/";
  }
  else if (code === 37) { // %
    this.tokenprio = 2;
    this.tokenindex = "%";
  }
  else if (code === 94) { // ^
    this.tokenprio = 3;
    this.tokenindex = "^";
  }
  else {
    return false;
  }
  this.pos++;
  return true;
},
```

```
isSign: function () {
  var code = this.expression.charCodeAt(this.pos - 1);
  if (code === 45 || code === 43) { // -
    return true;
  }
  return false;
},

isPositiveSign: function () {
  var code = this.expression.charCodeAt(this.pos - 1);
  if (code === 43) { // -
    return true;
  }
  return false;
},
isNegativeSign: function () {
  var code = this.expression.charCodeAt(this.pos - 1);
  if (code === 45) { // -
    return true;
  }
  return false;
},

isLeftParenth: function () {
  var code = this.expression.charCodeAt(this.pos);
  if (code === 40) { // (
    this.pos++;
    this.tmpprio += 10;
    return true;
  }
  return false;
},
isRightParenth: function () {
  var code = this.expression.charCodeAt(this.pos);
  if (code === 41) { // )
    this.pos++;
    this.tmpprio -= 10;
    return true;
  }
  return false;
},

isComma: function () {
  var code = this.expression.charCodeAt(this.pos);
  if (code === 44) { // ,
    this.pos++;
    this.tokenprio = -1;
    this.tokenindex = ",";
    return true;
  }
  return false;
},
isWhite: function () {
  var code = this.expression.charCodeAt(this.pos);
  if (code === 32 || code === 9 || code === 10 || code === 13) {
    this.pos++;
    return true;
  }
  return false;
},

isOp1: function () {
  var str = "";
  for (var i = this.pos; i < this.expression.length; i++) {
    var c = this.expression.charAt(i);
    if (c.toUpperCase() === c.toLowerCase()) {
      if (i === this.pos || c < '0' || c > '9') {
        break;
      }
    }
    str += c;
  }
  if (str.length > 0 && (str in this.ops1)) {
    this.tokenindex = str;
    this.tokenprio = 5;
    this.pos += str.length;
    return true;
  }
  return false;
},
```

```
    isOp2: function () {
      var str = "";
      for (var i = this.pos; i < this.expression.length; i++) {
        var c = this.expression.charAt(i);
        if (c.toUpperCase() === c.toLowerCase()) {
          if (i === this.pos || c < '0' || c > '9') {
            break;
          }
        }
        str += c;
      }
      if (str.length > 0 && (str in this.ops2)) {
        this.tokenindex = str;
        this.tokenprio = 5;
        this.pos += str.length;
        return true;
      }
      return false;
    },

    isVar: function () {
      var str = "";
      for (var i = this.pos; i < this.expression.length; i++) {
        var c = this.expression.charAt(i);
        if (c.toUpperCase() === c.toLowerCase()) {
          if (i === this.pos || c < '0' || c > '9') {
            break;
          }
        }
        str += c;
      }
      if (str.length > 0) {
        this.tokenindex = str;
        this.tokenprio = 4;
        this.pos += str.length;
        return true;
      }
      return false;
    },

    isComment: function () {
      var code = this.expression.charCodeAt(this.pos - 1);
      if (code === 47 && this.expression.charCodeAt(this.pos) === 42) {
        this.pos = this.expression.indexOf("*/", this.pos) + 2;
        if (this.pos === 1) {
          this.pos = this.expression.length;
        }
        return true;
      }
      return false;
    }
  };

  scope.Parser = Parser;
  return Parser
})(typeof exports === 'undefined' ? {} : exports);


// ---------------------------------------------------------------------------
// End of "parser.js"
// ---------------------------------------------------------------------------
```

● Verzeichnis aller Bibliotheks-Funktionen von „mathe.js“

```
var info, info1, info2, info3, info4;   // Fünf Stringvariable
var error = -999;                       // Numerische Error-Variable
var n = 3;                              // Numerische Dimensions-Variable
var L1xy;                               // Erster Lösungsvektor (n-Array)
var L2xy;                               // Zweiter Lösungsvektor (n-Array)
var mat;                                // mehrdimensionale (n)x(n+1)- Matrix

function myTrim(x)                      // Führende und folgende Blanks von x entfernen
function round2(x)                      // x auf 2 Dezimalen runden
function round4(x)                      // x auf 4 Dezimalen runden
function deg(x)                         // Umwandlung von Bogenmaß in Gradmaß
function rad(x)                         // Umwandlung von Gradmaß in Bogenmaß
function Max6(a,b,c,d,e,f)              // Maximum von 6 Zahlen
function Min6(a,b,c,d,e,f)              // Minimum von 6 Zahlen

function TPoint(tx, ty)                 // Konstruktor für das Punkt-Objekt (2-dim.)
function TPoint3(tx, ty, tz)            // Konstruktor für das Punkt-Objekt (3-dim.)

function Flaeche1(a,b,c)                // Fläche eines Dreiecks
function Winkel1(a,b,c)                 // Winkel w bei Eckpunkt B
function ggt(a,b)                       // größter gemeinsamer Teiler von a und b
function kgv(a,b)                       // kleinstes gemeinsames Vielfache von a und b
function Primfakt(x)                    // Primfaktoren einer Zahl x

function differ(u,x)
  // Erste Ableitung, u = Funktionsterm f(x), x = Argument

function differ2(u,x)
  // Zweite Ableitung, u = Funktionsterm f(x), x = Argument

function tangente(u,Q)
  // Tangente y = k*x + d in Punkt Q mit u als f(x): k = returnWert.x, d = returnWert.y

function nullst(u,a,b)
   // Nullstelle zwischen a und b (mit u als f(x))

function extrema(u,a,b)
   // Extremstelle zwischen a und b (mit u als f(x))

function wendpun(u,a,b)
   // Wendestelle zwischen a und b (mit u als f(x))

function integ(u,a,b)
   // bestimmtes Integral von a bis b (mit u als f(x))

function integbog(u,a,b)
   // Bogenlänge einer Kurve von a bis b (mit u als f(x)), Bogenlänge = returnWert

function integbogX(u,a,b)
   // x-Wert des Bogenschwerpunktes (S) einer Kurve von a bis b (mit u als f(x))
   // xS = returnWert / Bogenlänge

function integbogY(u,a,b)
   // y-Wert des Bogenschwerpunktes (S) einer Kurve von a bis b (mit u als f(x))
   // yS = returnWert / Bogenlänge

function integflaX(u,a,b)
   // x-Wert des Flächenschwerpunktes (S) einer Kurve von a bis b (mit u als f(x))
   // xS = returnWert / Fläche

function integflaY(u,a,b)
   // y-Wert des Flächenschwerpunktes (S) einer Kurve von a bis b (mit u als f(x))
   // yS = returnWert / Fläche

function integfla(u,a,b)
   // Oberfläche eines Drehkörpers von a bis b (mit u als f(x))
   // Oberfläche = returnWert

function integvol(u,a,b)
   // Volumen eines Drehkörpers von a bis b (mit u als f(x))
   // Volumen = returnWert

function integvolX(u,a,b)
   // x-Wert des Drehkörperschwerpunkt(S) einer Kurve von a bis b (mit u als f(x))
   // xS = returnWert / Volumen
```

```
function GER(P9,Q9)
   // Gerade durch die Punkte P9 und Q9, Ergebnisse in L1xy
   // Senkrechte Gerade bei "-999"
   // wenn k = ~, dann L1xy[0] = 1, L1xy[1] = x, L1xy[2] = -999
   // wenn k <> ~, dann L1xy[0] = 0, L1xy[1] = k, L1xy[2] = d
function SGG(A9,B9,C9,D9)
   // Schnittpunkt der Geraden g(A9,B9) und h(C9,D9)
   // wenn kg = ~, dann L1xy[0] = 1, L1xy[1] = x, L1xy[2] = -999
   // wenn kg <> ~, dann L1xy[0] = 0, L1xy[1] = k, L1xy[2] = d
   // wenn kh = ~, dann L2xy[0] = 1, L2xy[1] = x, L2xy[2] = -999
   // wenn kh <> ~, dann L2xy[0] = 0, L2xy[1] = k, L2xy[2] = d
   // globale Zusatzinformationen in info1, info2 und info3
function SKG(M9,r9,A9,B9)
   // Schnittpunkte der Geraden g(A9,B9) mit dem Kreis k(M9,r9)
   // Erster Schnittpunkt: (L1xy[1] / L2xy[1])
   // Zweiter Schnittpunkt: (L1xy[2] / L2xy[2])
   // KEINE Schnittpunkte bei L1xy[0] = 3
   // Geradengleichung in "info1"
function SKK(M11,r11,M22,r22)
  // Schnitt von Kreis und Kreis
  // Erster Schnittpunkt  S1(L1xy[1]/L2xy[1])
  // Zweiter Schnittpunkt S2(L1xy[2]/L2xy[2])
  // Fehlererkennung mit "-999"

function SSM(A,B)
   // Streckensymmetrale y = k*x+d von AB
   // mit k = L1xy[1] und d = L1xy[2]
   // und Streckenmittelpunkt L2xy[1] = M9.x und L2xy[2] = M9.y
   // Senkrechte Gerade bei "-999"
function WSM(P,Q,R)
   // Winkelsymmetrale y = k*x + d von w(PQR)
   // L1xy[1] = k und L1xy[2] = d
   // Senkrechte Gerade bei "-999"

function LOT(C,A,B)
   // Das Lot y = k*x + d von C auf AB
   // mit k = L1xy[1] und d = L1xy[2]
   // Lotfußpunkt C mit C.x = L2xy[1] und C.y = L2xy[2]
   // Senkrechte Gerade bei "-999"

function deta(b,n)
  // Berechnung der Determinante
  // b bzw. a = Koeffizienten-Matrix "mat[i][j]", n = Anzahl
function linglei(b,n,z)
  // Lösung eines linearen Gleichungsystems in n Variablen
  // b bzw. a = Koeffizienten-Matrix "mat[i][j]"
  // z bzw. x = lösungs-Vektor "L1xy[i]"

function quagl(a,b,c)
  // Lösungen einer quadratischen gleichung
  // Lösungsart L1xy[0], ( 1 = eine, 2 = zwei, 3 = keine )
  // Lösungen L1xy[1] und L1xy[2]

function vekmal(i,A)                  // Multiplikation eines Vektors mit einer Zahl (2D)
function veksum(A,B)                  // Addition zweier Vektoren (2D)
function veklin(s,t,A,B)              // Linearkombination zweier Vektoren (2D)
function Shap(P,Q)                    // Streckenhalbierungspunkt (2D)
function Spunkt(P,Q,R)                // Schwerpunkt (2D)
function Slen(P,Q)                    // Distanz zweier Punkte (2D)
function Umfang(P,Q,R)                // Umfang eines Dreiecks (2D)
function Flaeche(P,Q,R)               // Fläche eines Dreiecks (2D)
function Winkel(P,Q,R)                // Winkel w bei Punkt Q (2D)
function vekmul(A,B)                  // Skalarprodukt zweier Vektoren (2D)

function vekmal3(n,P3)                // Multiplikation eines Vektors mit einer Zahl (3D)
function veksum3(P3,Q3)               // Addition zweier Vektoren (3D)
function veklin3(s,t,P3,Q3)           // Linearkombination zweier Vektoren (3D)
function Shap3(P,Q)                   // Streckenhalbierungspunkt (3D)
function Spunkt3(P,Q,R)               // Schwerpunkt (3D)
function Slen3(P3,Q3)                 // Distanz zweier Punkte (3D)
function Umfang3(P3,Q3,R3)            // Umfang eines Dreiecks (3D)
function Flaeche3(P3,Q3,R3)           // Fläche eines Dreiecks (3D)
function Winkel3(P3,Q3,R3)            // Winkel w bei Punkt Q3 (3D)
function vekmul3(P3,Q3)               // Skalarprodukt zweier Vektoren (3D)
function vekkreuz3(P3,Q3)             // Kreuzprodukt zweier Vektoren (3D)
function vekspat3(P3,Q3,R3)           // Spatprodukt zweier Vektoren (3D)
```

● Verzeichnis aller Bibliotheks-Funktionen von „parser.js“

```
function fun(s,x)    // Funktionterm "s" in nur EINER Variablen x, returnWert = Funktionswert
function formel(ss)  // Formel "ss" in 26 Variablen a,b,c,...,y,z, returnWert = Funktionswert
```

Die Funktion „***fun(s,x)“*** dient der Programmierung von einfachen mathematischen Funktionen, wobei der erste Parameter der Funktionsterm ist und der zweite Parameter das Funktionsargument.

Die Funktion ***„formel(ss)“*** ermöglicht die Eingabe einer mathematischen Formel, welche die 26 Variablen ***a, b, c, . . ., x, y, z*** enthalten kann. Zurückgeliefert wird entweder eine relle Zahl oder bei fehlerhafter Eingabe „***NaN***" (Not a Number) bzw. "***Infinity***" (Unendlich).

Zulässige **Operatoren** sind: (,), +, -, *, /, ^, sqrt, % (=Rest), fac (=Faktorielle), round, rd2, rd4, floor, abs, exp, log, lg, deg, rad, sin, cos, tan, asin, acos, atan, atan2(y,x) (=Steigungswinkel), random(x) (=Zufallszahlen in [0,x)), PI (=Ludolfsche Zahl) und E (=Eulersche Zahl).

Beispiele: a % b, sin(rad(x)), deg(asin(x)), deg(atan2(a,b)), sqrt(x), log(x), exp(x), ……. Grundsätzlich werden alle Winkelfunktionen im Radianten-Maß berechnet. „deg(x)“ wandelt Radianten in Grade um. Hingegen wandelt „rad(x)“ Grade in Radianten um.

Programmieren mit JavaScript, Teil 5
FileReader, Fading, Drag & Drop

[5.1] Das FileReader-System

Das File API (Datei Application Programming Interface) von JavaScript enthält als zentrales Objekt das **FileReader**-Objekt, das verschiedene asynchrone Methoden zum Lesen von **File**- oder **Blob**-Objekten zur Verfügung stellt. Während mit **File** allgemein Dateien bezeichnet werden, sind **Blobs** binäre Objekte (Binary Large Objects), die beliebige Inhalte wie Texte und Bilder beinhalten können. Eine Instanz von FileReader wird erzeugt durch: **var reader = new FileReader();**

„readAsText()“

Die **readAsText()**-Methode wird zum Lesen von Textdateien verwendet. Sie besitzt zwei Parameter. Der erste Parameter bezeichnet das zu lesende Objekt (Textdatei). Der zweite, optionale Parameter bestimmt die Codierung (encoding) der Datei. Entfällt er, dann wird **UTF-8** automatisch verwendet. Für die deutsche Sprache mit ihren Umlauten gibt es auch die ältere Codetabelle „**ISO-8851-1**“.
Um eine Datei zu laden, wird eine Ereignis-Routine mittels „**eventListener**“ installiert, und wenn das **onload**-Ereignis ausgelöst wird, dann liefert das **result**-Merkmal von **FileReader** den Inhalt der Datei.

Das aufrufende HTML-Element ist das **<input>**-Element vom Typ „**file**“ mit der„**onchange**“-Funktion „**loadText()**“. Dadurch wird aus einem gewählten Ordner eine Dateiliste (array) „**file**“ erstellt, auf deren Einträge zugegriffen werden kann, wobei das erste Listenelement „**file[0]**“ ist. Weitere Attribute sind der Name (name), die Größe (size) und der Typ (type) von einer Datei.

Zum **Speichern von Dateien** kann ein **Download-Link** erzeugt werden, mit dessen Hilfe die jeweilige Datei in den Download-Ordner des Computers abgespeichert wird.

```
<DOCTYPE html>
<html>
<head>
<script>
   var fname = '';  // Dateiname

   function loadText() {
   // Textdatei laden
      var txtbox = document.getElementById('txtbox');
      var fileToLoad = document.getElementById('fileInput').files[0];
      var reader = new FileReader();
      textType = /text.*/;
      if ( !fileToLoad.type.match(textType) ) {
         document.getElementById('fileInput').value = ''; alert('Falscher Dateityp');  return;
      }
      reader.readAsText(fileToLoad);  // ecoding alternativ mit (fileToLoad,"ISO-8851-1")
      reader.addEventListener('load', function() {
         txtbox.value = reader.result;
      },false);
      fname = fileToLoad.name;
      alert(fname + ', ' + fileToLoad.size + ', ' + fileToLoad.type);
   }

   function saveText() {
   // Textdatei speichern
     var textFile = document.getElementById('txtbox').value;
     var textBlob = new Blob([textFile], {type:'text/plain'});
     var link = document.createElement('a');
     link.href = window.URL.createObjectURL(textBlob);
     fname = prompt('Dateiname',fname);
     link.download = fname;
     document.body.appendChild(link);
     link.click();
     document.body.removeChild(link);
   }
</script>
</head>

<body>
  <h3> Textdateien laden und speichern</h3>
  <p>Textbox zum Schreiben eines Textes:</p>
  <textarea id="txtbox" value="" cols="80" rows="20"></textarea>
  <br><br>
  <input type="file" id="fileInput" onchange="loadText()">
  <br><br>
  <input type="button" value="Text speichern" onclick="saveText()">
</body>
</html>
```

„readAsDataURL()"

Die **readAsDataURL()**-Methode wird zum Lesen von beliebigen Dateien als **Blobs** verwendet. Die Methode greift über eine **URL** (Uniform Ressource Locator) auf die Datei zu, welche dann beispielsweise dem **src**-Merkmal eines **Images** zugewiesen werden kann.

Um eine Datei zu laden, wird eine Ereignis-Routine mittels „**eventListener**" installiert, und wenn das **onload**-Ereignis ausgelöst wird, dann liefert das **result**-Merkmal von **FileReader** den Inhalt der Datei. Dieser Inhalt ist hier kein Text, sondern ein Blob, welcher den binären Code der Bilddatei enthält.

Das aufrufende HTML-Element ist das **<input>**-Element vom Typ „**file**" mit der„**onchange**"-Funktion „**loadImage()**".Dadurch wird aus einem gewählten Ordner eine Dateiliste (array) „**file**" erstellt, auf deren Einträge zugegriffen werden kann, wobei das erste Listenelement „**file[0]**" ist. Weitere Attribute sind der Name (name), die Größe (size) und der Typ (type) von einer Datei.

Zum **Speichern von Dateien** kann ein **Download-Link** erzeugt werden, mit dessen Hilfe die jeweilige Datei in den Download-Ordner des Computers abgespeichert wird

```
<DOCTYPE html>
<html>
<head>
<script>
   var fname = '';  // Dateiname

   function loadImage() {
   // Grafikdatei laden
     var imgbox = document.getElementById('picture');
     var fileToLoad = document.getElementById('fileInput').files[0];
     var reader = new FileReader();
     imageType = /image.*/;
     if ( !fileToLoad.type.match(imageType) ) {
        document.getElementById('fileInput').value = '';
        alert('Falscher Dateityp');
        return;
     }
     reader.readAsDataURL(fileToLoad);
     reader.addEventListener("load", function() {
        imgbox.src = reader.result;
     }, false);
     fname = fileToLoad.name;
     alert(fname + ', ' + fileToLoad.size + ', ' + fileToLoad.type);
   }

   function saveImage() {
   // Grafikdatei speichern
     document.getElementById('fileInput').value = '';
     grafik = document.getElementById('picture').src;
     var link = document.createElement('a');
     link.href = grafik;
     fname = prompt('Bildname',fname);
     link.download = fname;
     document.body.appendChild(link);
     link.click();
     document.body.removeChild(link);
   }
</script>
</head>

<body>
  <h3> Grafikdateien laden und speichern</h3>
  <img id="picture" src=" " style="height:400px"></img>
  <br><br>
  <input type="file" id="fileInput" onchange="loadImage()">
  <br><br>
  <input type="button" value="Grafik speichern" onclick="saveImage()">
  <br>
</body>
</html>
```

„readAsArrayBuffer()"

Diese dritte Methode zum Lesen von Dateien wird u.a. zum Lesen von Audios (z.B. mp3) und Videos (z.B. mp4) verwendet.

[5.1.1] „txtfile.html“

Ein Programm zum Laden und Speichern von Textdateien in ein Textfeld.


```
<!DOCTYPE html">
<html>
<head>
   <title>txtfile</title>
   <meta charset="ISO-8859-1">
   <meta name="description" content="Read & Write von Texten">
   <meta name="author" content="Herbert Paukert">

   <style type="text/css">
     body { background-color:#E0E8EF; color:black;
            font-family:sans-serif; font-size:16px; font-weight:normal;
            text-size-adjust: none; margin:3%;
     }
     #txtbox { font-family:sans-serif; font-size:14px; font-weight:normal;
               padding:10px; border:2px solid darkblue;
     }
   </style>

   <script>

   function loadText() {
   /* Textdatei laden */
      var txtbox = document.getElementById('txtbox');
      var fileToLoad = document.getElementById('fileInput').files[0];
      var reader = new FileReader();
      textType = /text.*/;
      if ( !fileToLoad.type.match(textType) ) {
         document.getElementById('fileInput').value = '';
         alert('Falscher Dateityp');
         return;
      }
      reader.readAsText(fileToLoad);
      reader.addEventListener('load', function() {
         txtbox.value = reader.result;
      },false);
   }
```

```
   function saveText() {
   /* Textdatei speichern */
     document.getElementById('fileInput').value = '';
     ext = '.txt';
     fname = 'text';
     textFile = document.getElementById('txtbox').value;
     var textBlob = new Blob([textFile], {type:'text/plain'});
     var link = document.createElement('a');
     link.href = window.URL.createObjectURL(textBlob);
     fname = prompt('Dateiname ohne Extension',fname);
     fname = fname + ext;
     link.download = fname;
     document.body.appendChild(link);
     link.click();
     document.body.removeChild(link);
   }

   </script>
</head>

<body>
  <h3> Textdateien laden und speichern (c) H.P.</h3>
  <p>Textbox zum Schreiben eines Textes:</p>
  <textarea id="txtbox" value="" cols="80" rows="20">

     Was immer auch geschieht,
     nie dürft ihr so tief sinken,
     um von dem Kakao,
     durch den man euch zieht,
     auch noch zu trinken.

  </textarea>
  <br><br>
  <input type="file" id="fileInput" onchange="loadText()">
  <br><br>
  <input type="button" value="Text speichern" onclick="saveText()">
  <br>
</body>
</html>
```

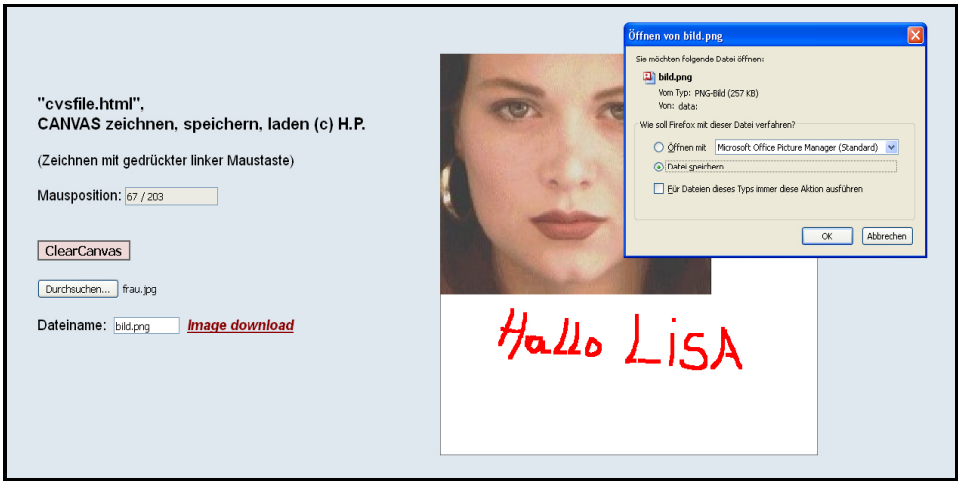
[5.1.2] „imgfile.html“

Ein Programm zum Laden und Speichern von Grafikdateien in ein Image.

```
<!doctype html>
<html>
<head>
   <title>imgfile</title>
   <meta charset="ISO-8859-1">
   <meta name="description" content="Read & Write von Images">
   <meta name="author" content="Herbert Paukert">

   <style type="text/css">
     body{background-color:#E0E8EF; color:black;
     font-family:sans-serif; font-size:18px; font-weight:normal;
     text-size-adjust: none; margin:3%;}
   </style>

   <script>

   function loadImage() {
   /* Grafikdatei laden */
     var imgbox = document.getElementById('pic');
     var fileToLoad = document.getElementById('fileInput').files[0];
     var reader = new FileReader();
     imageType = /image.*/;
     if ( !fileToLoad.type.match(imageType) ) {
        document.getElementById('fileInput').value = '';
        alert('Falscher Dateityp');
        return;
     }
     reader.readAsDataURL(fileToLoad);
     reader.addEventListener("load", function () {
        imgbox.src = reader.result;
     }, false);
   }

  function saveImage() {
  /* Grafikdatei speichern */
     document.getElementById('fileInput').value = '';
     ext = '.jpg';
     fname = 'bild';
     grafik = document.getElementById('pic').src;
     var link = document.createElement('a');
     link.href = grafik;
     if ( grafik.indexOf('image/png') > -1 )  { ext = '.png' }
     if ( grafik.indexOf('image/jpeg') > -1 ) { ext = '.jpg' }
     fname = fname + ext;
     fname = prompt('Bildname (.png, .jpg)',fname);
     link.download = fname;   /* grafik; */
     document.body.appendChild(link);
     link.click();
     document.body.removeChild(link);
  }

</script>
</head>

<body>
  <h3> Grafikdateien laden und speichern (c) H.P.</h3>
  <img id="pic" src=" " style="height:400px">
  <br><br>
  <input type="file" id="fileInput" onchange="loadImage()">
  <br><br>
  <input type="button" value="Grafik speichern" onclick="saveImage()">
</body>
</html>
```

[5.1.3] „cvsfile.html“

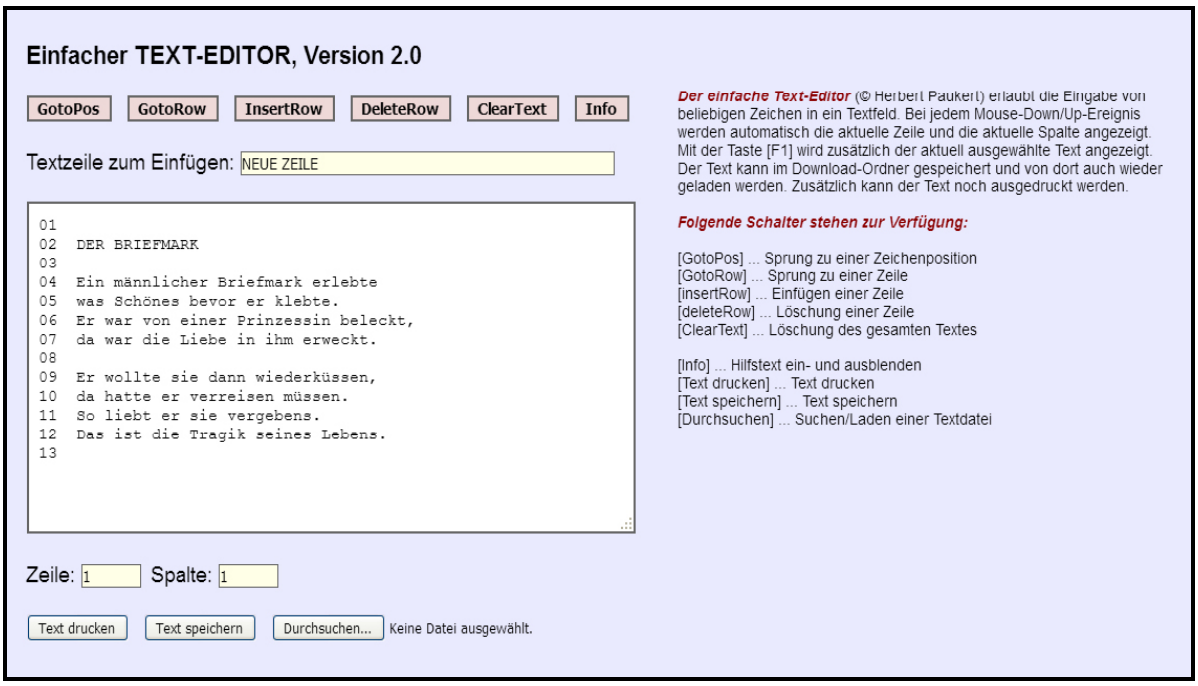
Ein Programm zum Zeichnen, Laden und Speichern von Grafiken in einem Canvas.




```
<!doctype html>
<html>
<head>
  <title> cvsfile </title>
  <meta charset="ISO-8859-1">
  <meta name="description" content="Read & Write von Canvas">
  <meta name="author" content="Herbert Paukert">

  <style>
     body{background-color:#E0E8EF; color:black;
       font-family:sans-serif; font-size:18px; font-weight:normal; text-size-adjust: none;
       text-align:left; margin:5%;
     }
     .btn {border:2px solid #666666; background-color:#EED8D8; font-size:16px;
       font-weight:bold;}
     .lnk {color:darkred; font-style:italic; font-weight:bold; }
  </style>

  <script>
    var canvas;               /* globale Canvas-Variable */
    var ctx;                  /* globale Kontext-Variable als Schnittstelle zum Canvas */
    var x,y;
    var go = false;

    function initCanvas() {
    /* Canvas initialisieren */
       canvas = document.getElementById('MyCanvas');
       ctx = canvas.getContext('2d');
       ctx.lineWidth = 1;
       ctx.fillStyle = "#F8F8F4";
       ctx.fillRect(0,0,500,500);
       ctx.strokeStyle = "black";
       ctx.strokeRect(0,0,500,500);
    }

    function paint() {
    /* Zeichnen ausführen */
       if (go) {
         ctx.fillStyle = "red";
         ctx.fillRect(x,y,5,5);
       }
    }

    function down() { go = true; }
    function up()   { go = false; }

    function move(ev) {
       x = ev.clientX - 600;
       y = ev.clientY - 60;
       var info = x + ' / ' + y;
       document.MyForm.aus.value = info;
       paint();
    }
```

```
    function loadFile() {
    /* Eine Grafikdatei angepasst in den Canvas laden */
       var img = new Image();
       var fileToLoad = document.getElementById('fileInput').files[0];
       var reader = new FileReader();

       imageType = /image./;
       if ( !fileToLoad.type.match(imageType) ) {
          document.getElementById('fileInput').value = '';
          alert('Falscher Dateityp');
          return;
       }
       reader.readAsDataURL(fileToLoad);
       reader.addEventListener("load", function() {
          img.src = reader.result;
       }, false);

       img.onload = function() {
         var canvas = document.getElementById('MyCanvas');
         var ctx = canvas.getContext('2d');

         /* Image-Anpassung an den Canvas-Behälter (resizing) */

         if ( img.width > img.height) {
           if ( img.width > canvas.width ) {
             img.height = img.height * canvas.width / img.width;
             img.width  = canvas.width;
           }
         }
         else {
           if ( img.width > canvas.width ) {
             img.width  = img.width * canvas.height / img.height;
             img.height = canvas.height;
           }
         }

         ctx.drawImage(img,0,0,img.width, img.height);
       }
    }

    function saveFile() {
    /* Den Canvas in eine Grafikdatei speichern */
       document.getElementById('fileInput').value = '';
       function downloadCanvas(link, canvasId) {
         link.href = document.getElementById(canvasId).toDataURL();
         link.download = document.getElementById('ein').value;
       }
       document.getElementById('downld').addEventListener('click', function() {
         downloadCanvas(this, 'MyCanvas'); }, false);
    }

   </script>
</head>

<body onload = initCanvas()>
<div style="position:absolute; top:60px; left:600px">
  <canvas id='MyCanvas' width='500' height='500'
     onmousedown=down() onmouseup=up() onmousemove=move(event)>
     Your browser does not support HTML5 Canvas.
  </canvas>
</div>

<form name="MyForm">
<br>
<h3>CANVAS zeichnen, speichern, laden (c) H.P.</h3>
(Zeichnen mit gedrückter linker Maustaste)
<br><br>
Mausposition: <input id="aus" type="text" value=" " size="16" readonly>
<br><br><br>
<input type="button" value="ClearCanvas" class="btn" onClick="initCanvas();"><br><br>
<input type="file" id="fileInput" onchange="loadFile()"><br><br>
Dateiname: 
<input id="ein" type="text" value="bild.png" size="10"> 
<a id="downld" class="lnk" onclick="saveFile()">Image download</a>
<br>
</form>
</body>
</html>
```

[5.1.4] Einfacher Text-Editor




```
<!DOCTYPE html>
<html>
<head>
<title>Text-Editor</title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Text-Editor">

<style>
body  {background-color:#EAEAFF; color:black; font-family:Arial;
       font-size:19px; font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
#edit {border: 2px solid #666666; padding: 10px; font-family:Courier New; font-size:15px;}
#help {position: absolute; top: 70px; left: 700px; visibility: hidden; font-family:Arial;
       font-size:14px;}
.cd1  {border: 1px solid #666666; background-color: #FFFFE8; font-size: 14px;}
.cd2  {border: 2px solid #666666; background-color: #EED8D8; font-size: 14px; font-weight: bold;}
</style>

<script>
  var hide = true;                                        // Info on/off
  var flag = false;

  function KeyListener(ev) {
  /* Tastatur-Eventhandler */
    var taste = ev.keyCode || ev.which;
    //  alert('Tastencode = ' + taste);
    if (taste == 112) { flag = true;  check(); }          // Funktionstaste F1
  }

  function check() {
  /* Ermittelt die aktuelle Cursorposition (row,col) im Text. */
  /* Ermittelt auch die aktuelle markierte Textzeile */
    const LF = '\n';                                      // LineFeed
    var el  = document.getElementById("edit");            // Textarea
    var tex = el.value;                                   // Gesamter Text
    var len = el.value.length;                            // Gesamte Textlänge
    var tar0 = tex.split(LF);                             // Text -> Array
    var countLF = tar0.length;                            // Gesamte Zeilenanzahl
    var startPos  = el.selectionStart;                    // Auswahl-Start
    var endPos    = el.selectionEnd;                      // Auswahl-Ende
    var selLength = endPos - startPos;                    // Auswahl-Länge
    var selText   = tex.substring(startPos,endPos);       // Auswahl-Text
    if ( flag ) { alert(selText); flag = false; return; }
```

```
  var texTo   = tex.substring(0,startPos);               // Text von 0 bis Start
  var tar1    = texTo.split(LF);                         // Text -> Array
  var rowPos  = tar1.length;                             // aktuelle Zeilenposition
  rowText = tar0[rowPos-1];                              // aktueller Zeilentext
  var lastLF  = texTo.lastIndexOf(LF);                   // Letzter Linefeed im Text
  var colPos  = startPos - lastLF;                       // aktuelle Spaltenposition
  document.MyForm.aus2.value = rowPos;                   // Zeilenposition anzeigen
  document.MyForm.aus3.value = colPos;                   // Spaltenposition anzeigen
}

function loadText() {
/* Textdatei laden */
  var txtbox = document.getElementById('edit');
  var fileToLoad = document.getElementById('fileInput').files[0];
  var reader = new FileReader();
  textType = /text.*/;
  if ( !fileToLoad.type.match(textType) ) {
    document.getElementById('fileInput').value = '';
    alert('Falscher Dateityp');
    return;
  }
  reader.readAsText(fileToLoad);
  reader.addEventListener('load', function() {
    txtbox.value = reader.result;
  },false);
}

function saveText() {
/* Textdatei speichern */
  document.getElementById('fileInput').value = '';
  ext = '.txt';
  fname = 'text';
  textFile = document.getElementById('edit').value;
  var textBlob = new Blob([textFile], {type:'text/plain'});
  var link = document.createElement('a');
  link.href = window.URL.createObjectURL(textBlob);
  fname = prompt('Dateiname ohne Extension',fname);
  fname = fname + ext;
  link.download = fname;
  document.body.appendChild(link);
  link.click();
  document.body.removeChild(link);
}

function getCaretPos(el) {
/* die aktuelle, absolute Cursorposition ermitteln */
  var pos;
  if (document.selection) {
    el.focus();
    var sel = document.selection.createRange();
    sel.moveStart('character', -el.value.length);
    pos = sel.text.length;
  }
  else if (el.selectionStart || el.selectionStart == '0') {
    pos = el.selectionStart;
  }
  return pos;
}

function setCaretPos(el, pos) {
/* zu einer absoluten Cursorposition springen */
  if (el.setSelectionRange) {
    el.focus();
    el.setSelectionRange(pos, pos);
  }
  else if (el.createTextRange) {
    var range = el.createTextRange();
    range.collapse(true);
    range.moveEnd('character', pos);
    range.moveStart('character', pos);
    range.select();
  }
}
```

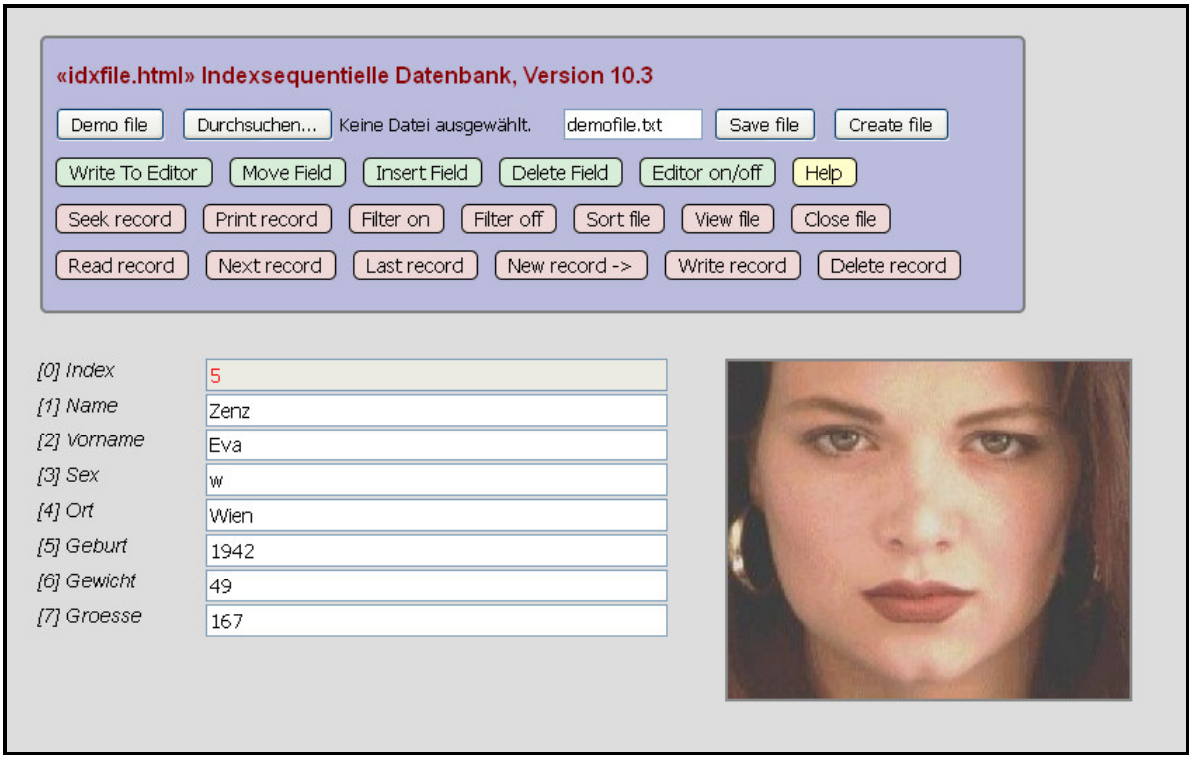
```
function gotoLine(el, row) {
/* zu einer Zeile springen */
  document.MyForm.aus2.value = row;
  document.MyForm.aus3.value = 1;
  const LF = '\n';
  var tex = el.value;
  var tar0 = tex.split(LF);
  var countLF = tar0.length;
  row = row - 1;
  if (row < 0) { row = 0; }
  if (row >= countLF-1) { row = countLF-1; }
  var count = 0;
  for (i = 0; i < row; i++) { count = count + tar0[i].length + 1; }
  setCaretPos(el, count);
}

function gotoRow() {
/* zu einer Zeile springen */
  var row = 1;
  row = prompt('RowNumber',row);
  gotoLine(document.getElementById('edit'), row);
}

function gotoPos() {
/* zu einer absoluten Position springen */
  var pos = getCaretPos(document.getElementById('edit'));
  pos = prompt('PosNumber',pos);
  setCaretPos(document.getElementById('edit'), pos)
}

function insertRow() {
/* eine Zeile einfügen */
  el = document.getElementById('edit');
  var tex = el.value;
  var intex = document.MyForm.aus1.value;
  var row = 1;
  row = prompt('RowNumber',row);
  gotoLine(el, row);
  startPos  = el.selectionStart;
  var tex1 = tex.substring(0,startPos) + intex + '\n' + tex.substring(startPos);
  document.MyForm.edit.value = tex1;
  setCaretPos(el, startPos)
}

function deleteRow() {
/* eine Zeile löschen */
  el = document.getElementById('edit');
  var tex = el.value;
  var row = 1;
  row = prompt('RowNumber',row);
  gotoLine(el, row);
  var startPos  = el.selectionStart;
  var tex1 = tex.substring(0,startPos);
  var tex2 = tex.substring(startPos);
  endPos = tex2.indexOf('\n');
  tex3 = tex2.slice(endPos+1);
  document.MyForm.edit.value = tex1 + tex3;
  setCaretPos(el, startPos)
}

function clearText() {
/* Text und Grafik löschen */
  document.MyForm.aus1.value = '';
  document.MyForm.aus2.value = '';
  document.MyForm.aus3.value = '';
  document.MyForm.edit.value = '';
  document.MyForm.edit.focus();
}

function printText() {
/* Text drucken */
  var text = document.MyForm.edit.value;
  w = window.open(","location=no,scrollbars=no,menubar=no");
  w.document.open();
  w.document.write("<html><body><pre>");
  w.document.write(text);
  w.document.write("</pre></html></body>");
  w.document.close();
}
```

```
  function Info() {
  /* Hilfstext ein-/ausblenden */
    hide = !hide;
    if (hide == true) { document.getElementById("help").style.visibility = 'hidden';  }
    else { document.getElementById("help").style.visibility = 'visible'; }
    document.MyForm.edit.focus();
  }
</script>
</head>

<body onmousedown = "check()" onmouseup = "check()" onkeydown = "KeyListener(event)" >

<form name="MyForm">
<h3>Einfacher TEXT-EDITOR, Version 2.0</h3>
<input type="button" name="btn" value="GotoPos" class="cd2" onclick="gotoPos()">  
<input type="button" name="btn" value="GotoRow" class="cd2" onclick="gotoRow()">  
<input type="button" name="btn" value="InsertRow" class="cd2" onclick="insertRow()">  
<input type="button" name="btn" value="DeleteRow" class="cd2" onclick="deleteRow()">  
<input type="button" name="btn" value="ClearText" class="cd2" onclick="clearText()">  
<input type="button" name="btn" value="Info" class="cd2" onclick="Info()">
<br><br>
Textzeile zum Einfügen: <input type="text" name="aus1" value="NEUE ZEILE" size="55" id="aus1"
class="cd1">
<br><br>

<textarea name="edit" value="" id="edit" cols="60" rows="15" wrap="off">
01
02  DER BRIEFMARK
03
04  Ein männlicher Briefmark erlebte
05  was Schönes bevor er klebte.
06  Er war von einer Prinzessin beleckt,
07  da war die Liebe in ihm erweckt.
08
09  Er wollte sie dann wiederküssen,
10  da hatte er verreisen müssen.
11  So liebt er sie vergebens.
12  Das ist die Tragik seines Lebens.
13
</textarea>

<br><br>
Zeile: <input type="text" name="aus2" value="" size="5" id="aus2" class="cd1" readonly> 
Spalte: <input type="text" name="aus3" value="" size="5" id="aus3" class="cd1" readonly>
<br><br>
<input type="button" value="Text drucken" onclick="printText()">  
<input type="button" value="Text speichern" onclick="saveText()">  
<input type="file" id="fileInput" onchange="loadText()">  
<br>
<div id='help'>
<font color = "black">
<font color = "darkred"><b><i>Der einfache Text-Editor </b></i></font>
(&copy Herbert Paukert) erlaubt die Eingabe von<br>
beliebigen Zeichen in ein Textfeld. Bei jedem Mouse-Down/Up-Ereignis<br>
werden automatisch die aktuelle Zeile und die aktuelle Spalte angezeigt.<br>
Mit der Taste [F1] wird zusätzlich der aktuell ausgewählte Text angezeigt.<br>
Der Text kann im Download-Ordner gespeichert und von dort auch wieder<br>
geladen werden. Zusätzlich kann der Text noch ausgedruckt werden.<br>
<br>
<font color = "darkred">
<b><i>
Folgende Schalter stehen zur Verfügung:<br>
</b></i>
</font>
<br>
[GotoPos] ... Sprung zu einer Zeichenposition<br>
[GotoRow] ... Sprung zu einer Zeile<br>
[insertRow] ... Einfügen einer Zeile<br>
[deleteRow] ... Löschung einer Zeile<br>
[ClearText] ... Löschung des gesamten Textes<br>
<br>
[Info] ... Hilfstext ein- und ausblenden<br>
[Text drucken] ... Text drucken<br>
[Text speichern] ... Text speichern<br>
[Durchsuchen] ... Suchen/Laden einer Textdatei<br>
</font>
</div>
</form>
</body>
</html>
```

[5.2] Indexsequentielle Datenbank (idxfile.html)

Eine solche Datenbank (File) ist eine einfache Textdatei, die aus verschiedenen Datensätzen (Records) besteht, welche durch den Separator "#" getrennt sind. Jeder Datensatz enthält gleichviele Datenfelder (Fields), welche durch den Separator ";" getrennt sind. Die Datensätze sind automatisch fortlaufend nummeriert (indiziert). Der erste Datensatz (Header) enthält immer die Namen der Datenfelder. Das erste Datenfeld (0) enthät immer den Index, der nicht verändert werden kann. Das nachfolgende Datenbank-Beispiel "demofile.txt" besteht aus dem Header und 25 Datensätzen.

```
Index;Name;Vorname;Sex;Ort;Geburt;Gewicht;Groesse#
 1;Ronka;Lisa;w;Wien;1945;65;165#
 2;Meier;Herbert;m;Graz;1940;87;190#
 3;Dorfer;Maria;w;Graz;1950;83;160#
 4;Stanka;Rudolf;m;Linz;1943;61;171#
 5;Zenz;Eva;w;Wien;1942;49;167#
 6;Wille;Heinz;m;Linz;1940;82;182#
 7;Rutger;Herbert;m;Wien;1943;74;178#
 8;Hauer;Friedl;m;Linz;1942;76;178#
 9;Mueller;Roland;m;Wien;1941;81;185#
10;Wollner;Christa;w;Linz;1947;67;156#
11;Klaus;Eva;w;Wien;1948;58;160#
12;Holler;Sabine;w;Wien;1945;58;170#
13;Ebner;Harald;m;Graz;1940;78;178#
14;Bauer;Ernst;m;Graz;1945;72;178#
15;Kurz;Werner;m;Wien;1942;73;185#
16;Stadler;Susi;w;Linz;1950;56;169#
17;Wolf;Maria;w;Graz;1949;58;167#
18;Klad;Eva;w;Linz;1943;52;165#
19;Artner;Heinz;m;Wien;1943;61;173#
20;Baier;Helene;w;Wien;1944;65;171#
21;Adam;Franz;m;Wien;1945;90;177#
22;Neuner;Eva;w;Graz;1950;56;170#
23;Troll;Rudolf;m;Linz;1943;69;178#
24;Sarg;Willi;m;Wien;1950;93;182#
25;Toth;Maria;w;Wien;1947;61;157
```



[***Demo file***] lädt eine Demo-Datenbank aus dem Hauptspeicher.
[***Durchsuchen (browse)***] lädt eine Datenbank (z.B. myfile.txt) aus einem Ordner.
[***Save file***] speichert eine Datenbank immer in den Download-Ordner.

Erzeugung einer neuen Datenbank im Hauptspeicher:
Zuerst den internen Texteditor öffnen.
([***Edit on/off***] öffnet und schließt den Editor.)
Dann die Datensätze in den geöffneten Editor schreiben.
Zuletzt den Schalter [***Create file***] anklicken.

Änderung der Datenbank-Struktur im Hauptspeicher:
[***WriteTo Edit***] transferiert die aktuelle Datenbank in den Editor.
Mit [***Move Field***] wird ein Datenfeld verschoben.
Mit [***Insert Field***] wird ein neues Datenfeld eingefügt.
Mit [***Delete Field***] wird ein Datenfeld entfernt.
[***Create file***] erzeugt die geänderte Datenbank.

[***Read record***] zeigt einen wählbaren Datensatz.
[***Next record***] zeigt den nachfolgenden Datensatz.
[***Last record***] zeigt den vorangehenden Datensatz.
[***New record***] legt einen neuen Datensatz an,
dessen Inhalt dann mit [***Write record***] gespeichert wird.
[***Write record***] speichert einen Datensatz ab,
dessen Inhalt vorher geändert oder neu angelegt wurde.
[***Delete record***] löscht den aktuellen Datensatz,
wobei alle Datensätze neu nummeriert (indiziert) werden.
[***Seek record***] sucht in allen Datensätzen
in einem wählbaren Datenfeld nach einem
Suchbegriff (unabhängig von Groß- oder Klein-
Schreibung und immer am Feldanfang beginnend).
[***Print record***] druckt einen Datensatz aus.

[***Filter on***] filtert Datensätzen heraus. Dabei wird
zuerst ein wählbares Datenfeld eingegeben,
dann ein Vergleichsoperator (<, =, >, ?) und
zuletzt ein Suchbegriff. (?) bedeutet, dass der
Suchbegriff im Datenfeld enthalten sein muss.
Auch wiederholte Filterungen sind möglich.
Es erfolgt eine automatische Unterscheidung von
numerischen und nicht numerischen Datenfeldern.
[***Filter off***] löscht alle vorher gesetzten Filter.
Für einige Funktionen ([***New record***], usw.)
müssen alle gesetzten Filter gelöscht werden.

[***View file***] zeigt wählbare Datenfelder der gesamten Datenbank an
und ermöglicht auch deren statistische Auswertung.
[***Sort file***] sortiert die Datenbank nach einem Datenfeld.
Dabei erfolgt eine automatische Unterscheidung von
numerischen und nicht numerischen Datenfeldern.
[***Close file***] schließt die aktuelle Datenbank.

Ein Datenbank-Beispiel mit Header und drei Datensätzen:

```
Index; Name; Vorname; Sex;Telefon #
1; Zenz; Eva; w; 01/7234219 #
2; Meier; Fritz; m; 01/6157730 #
3; Adam; Franz; m; 01/2315643
```

Eine solche Textdatei kann mit jedem Texteditor erzeugt werden,
aber auch im internen Editor (mit [***Edit on/off***] und [***Create file***]).

Befinden sich im aktuellen Ordner JPG-Bilder mit den Dateinamen
"Feld1_Feld2.jpg", dann werden die Bilder bei den entsprechenden
Datensätzen automatisch angezeigt (beispielsweise "Zenz_Eva.jpg“).

Jede im Hauptspeicher erzeugte Datenbank kann mit [***Save file***] nur
im Download-Ordner abgespeichert werden (wegen Internet-Sicherheit).
Sie kann dann – falls gewünscht – im jeweiligen Betriebssytem vom
Download-Ordner in jeden anderen Ordner kopiert werden.

Hinweis: In die Datenfelder dürfen keine Steuerzeichen (, ; #)
geschrieben werden, d.h. keine Beistriche, Strichpunkte und Rauten,
weil diese Zeichen für die Struktur der Datensätze intern verwendet
werden.

Das Programm "idxfile.html" stellt menügeführt alle wichtigen Verwaltungsroutinen
für Datenbanken zur Verfügung:

- Laden, Speichern und Schließen von Datenbanken (*file management*).
- Eingabe, Ausgabe und Drucken von Datensätzen in **Maskenform**.
- Ändern, Löschen und Anfügen von Datensätzen (*edit data*).
- Ändern der Datenbank-Struktur (*modify structure*)
  durch Verschieben, Einfügen oder Entfernen von Datenfeldern
  im internen Texteditor. In der **Listenform** im Editor können
  Feldinhalte geändert und auch ganze Datensätze gelöscht werden.
- Suchen von Suchbegriffen in allen Datensätzen (*find data*).
- Mehrstufiges Filtern von Datensätzen (*select data)*.
- Sortieren der Datensätze entsprechend einem wählbaren Datenfeld (*sort data*).
- Listendarstellung von wählbaren Datenfeldern (*view data*).
  Zusätzlich kann daraus ein Datenfeld bestimmt werden,
  welches dann statistisch ausgewertet wird.

Die unten stehende Grafik zeigt die zu „demofile1.txt" geänderte Datenbank „demofile.txt",
wobei folgende sechs Routinefunktionen durchgeführt wurden:

(1) Laden der Datenbank „demofile.txt" (mit [Demo file]).

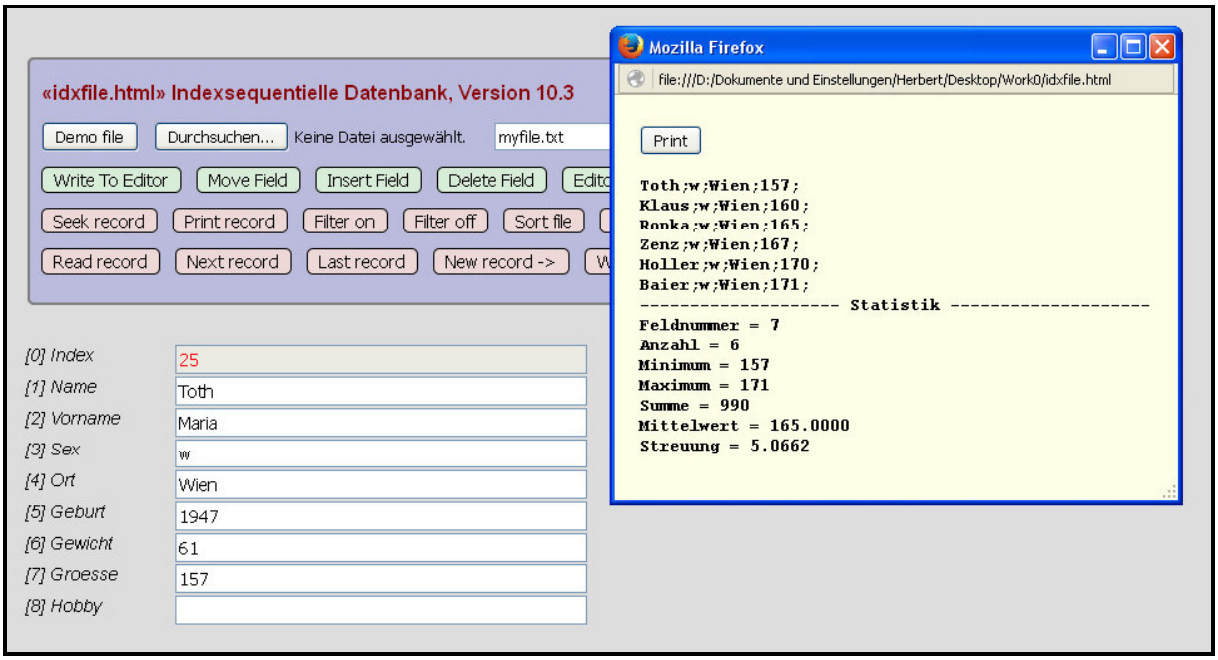
(2) Anhängen eines neuen Datenfeldes „Hobby" (mit [WriteTo Edit], [Insert Field] und [Create file]).

(3) Erster Datenfilter „Ort = Wien" (mit [Filter on]) → filtert 12 Datensätze (von 25).

(4) Zweiter Datenfilter „Sex = w" (mit [Filter on]). → filtert 6 Datensätze (von 12).

(5) Sortieren der Datensätze entsprechend dem Datenfeld „Groesse" (mit [Sort file]).

(6) Listendarstellung der Felder „Name", „Sex", „Ort" und „Groesse" (mit [View file]),
und statistische Auswertung des Datenfeldes „Groesse".



```
<!DOCTYPE html">
<html>
<head>
<title>Indexsequentielle Datenbank</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Indexsequentielle Datenbank">
<meta name="author" content="Herbert Paukert">

<style>
body {margin: 30px; background-color:#DDDDDD; color:black; font-family: Arial; font-size: 15px; }
.bs  {border: 1px solid black; border-radius:5px; background-color: #EED8D8; font-size: 15px;}
.bs1 {border: 1px solid black; border-radius:5px; background-color: #D8EED8; font-size: 15px;}
.bs2 {border: 1px solid black; border-radius:2px; background-color: #FFFFFF; font-size: 15px;}
.bs3 {border: 1px solid black; border-radius:5px; background-color: #FFFFCC; font-size: 15px;}
img  {position:absolute; left:540px; height:240px; border: 2px solid #888888; visibility:hidden;}
#tbox {border: 2px solid #888888; visibility:hidden; font-size: 14px; }
#tit {font-size:16px; font-weight: bold; color: darkred;}
#fs  {background-color:#BBBBDD; width:720px; border: 2px solid gray; border-radius:5px;}
#btns {line-height:8px; }
</style>

<script>
   var childWindow;
   var child = false;
   window.addEventListener( "unload", function() { closeChild(); } );

   function closeChild() {
   // Hilfetext schließen
     if (!childWindow || childWindow.closed) { return; }
     if (child) { childWindow.document.close(); childWindow.close(); }
   }

  function help_() {
  // Hilfetext anzeigen
    if (child) { closeChild(); child = false; return; }
    child = true;
    childWindow = window.open('','childWindow','top=20, left=720, width=600, height=640, srollbars=yes, menubar=no, toolbar=no');
    childWindow.document.open();
    childWindow.document.write('<html><head></head><body style="background-color:#FFFFE8; font-family:Arial, sans-serif;
                           font-size:13px; margin-left:20px; margin-bottom:40px;">');
    childWindow.document.write('<br><input type="button" value="Print" onclick="print();"><br><br>');
    childWindow.document.write('<b>Indexsequentielle Datenbank "idxfile.html" (c) Herbert Paukert.</b><br>');
    childWindow.document.write('<br>');
    childWindow.document.write('Eine solche Datenbank (file) ist eine einfache Textdatei, die aus<br>');
    childWindow.document.write('verschiedenen Datensätzen (records) besteht, welche durch den<br>');
    childWindow.document.write('Separator "#" getrennt sind. Jeder Datensatz enthält gleichviele<br>');
    childWindow.document.write('Datenfelder (fields), welche durch den Separator ";" getrennt sind.<br>');
    childWindow.document.write('Die Datensätze sind automatisch fortlaufend nummeriert (indiziert).<br>');
    childWindow.document.write('Der erste Datensatz (Header) enthält immer die Namen der Datenfelder.<br>');
    childWindow.document.write('Im ersten Datenfeld (0) ist der Index, der nicht verändert werden kann.<br>');
    childWindow.document.write('<br>');
    childWindow.document.write('Hinweis: In die Datenfelder dürfen keine Steuerzeichen<b>  ","  ";"  "#"</b><br>');
    childWindow.document.write('geschrieben werden, d.h. keine Beistriche, Strichpunkte und Rauten!<br>');
    childWindow.document.write('<br>');
    childWindow.document.write('<b>[Demo file]</b> lädt eine Demo-Datenbank aus dem Hauptspeicher.<br>');
    childWindow.document.write('<b>[Durchsuchen (browse)]</b> lädt eine Datenbank aus einem Ordner.<br>');
    childWindow.document.write('<b>[Save file]</b> speichert eine Datenbank immer in den Download-Ordner.<br>');
    childWindow.document.write('<br>');
    childWindow.document.write('Erzeugung einer neuen Datenbank im Hauptspeicher:<br>');
    childWindow.document.write('    Zuerst den internen Texteditor öffnen.<br>');
    childWindow.document.write('    <b>[Editor on/off]</b> öffnet oder schließt den Editor.<br>');
    childWindow.document.write('    Dann die Datensätze in den offenen Editor schreiben.<br>');
    childWindow.document.write('    Zuletzt den Schalter <b>[Create file]</b> anklicken.<br>');
    childWindow.document.write('<br>');
    childWindow.document.write('Änderung der Datenbank-Struktur im Hauptspeicher:<br>');
    childWindow.document.write('    <b>[WriteTo Editor]</b> transferiert die aktuelle Datenbank in den Editor.<br>');
    childWindow.document.write('    Mit <b>[Move Field]</b> wird ein Datenfeld verschoben.<br>');
    childWindow.document.write('    Mit <b>[Insert Field]</b> wird ein neues Datenfeld eingefügt.<br>');
    childWindow.document.write('    Mit <b>[Delete Field]</b> wird ein Datenfeld entfernt.<br>');
    childWindow.document.write('    <b>[Create file]</b> erzeugt die geänderte Datenbank.<br>');
    childWindow.document.write('<br>');
    childWindow.document.write('<b>[Read record]</b> zeigt einen wählbaren Datensatz.<br>');
    childWindow.document.write('<b>[Next record]</b> zeigt den nachfolgenden Datensatz.<br>');
    childWindow.document.write('<b>[Last record]</b> zeigt den vorangehenden Datensatz.<br>');
    childWindow.document.write('<b>[New record]</b> legt einen neuen Datensatz an,<br>');
    childWindow.document.write('    dessen Inhalt dann mit <b>[Write record]</b> gespeichert wird,<br>');
    childWindow.document.write('    wobei alle Datensätze neu nummeriert (indiziert) werden.<br>');
    childWindow.document.write('<b>[Write record]</b> speichert einen Datensatz ab,<br>');
    childWindow.document.write('    dessen Inhalt vorher geändert oder neu angelegt wurde.<br>');
    childWindow.document.write('<b>[Delete record]</b> löscht den aktuellen Datensatz,<br>');
    childWindow.document.write('    wobei alle Datensätze neu nummeriert (indiziert) werden.<br>');
    childWindow.document.write('<b>[Seek record]</b> sucht in allen Datensätzen<br>');
    childWindow.document.write('    in einem wählbaren Datenfeld nach einem<br>');
    childWindow.document.write('    Suchbegriff (unabhängig von Groß- oder Klein-<br>');
    childWindow.document.write('    Schreibung und immer am Feldanfang beginnend).<br>');
    childWindow.document.write('<b>[Print record]</b> druckt einen Datensatz aus.<br>');
    childWindow.document.write('<br>');
    childWindow.document.write('<b>[<i>Filter on</i>]</b> filtert Datensätze heraus. Dabei wird<br>');
    childWindow.document.write('    zuerst ein wählbares Datenfeld eingegeben,<br>');
    childWindow.document.write('    dann ein Vergleichsoperator (<, =, >, ?) und<br>');
    childWindow.document.write('    zuletzt ein Suchbegriff. (?) bedeutet, dass der<br>');
    childWindow.document.write('    Suchbegriff im Datenfeld enthalten sein muss.<br>');
    childWindow.document.write('    Auch wiederholte Filterungen sind möglich.<br>');
    childWindow.document.write('    Es erfolgt eine automatische Unterscheidung von<br>');
    childWindow.document.write('    numerischen und nicht numerischen Datenfeldern.<br>');
    childWindow.document.write('<b>[Filter off]</b> löscht alle vorher gesetzten Filter.<br>');
    childWindow.document.write('    Für einige Funktionen (<b>[New record]</b>,usw.) müssen<br>');
    childWindow.document.write('    alle gesetzten Filter gelöscht werden.<br>');
    childWindow.document.write('<br>');
```

```
    childWindow.document.write('<b>[Sort file]</b> sortiert die Datenbank nach einem Datenfeld.<br>');
    childWindow.document.write('    Dabei erfolgt eine automatische Unterscheidung von<br>');
    childWindow.document.write('    numerischen und nicht numerischen Datenfeldern.<br>');
    childWindow.document.write('<b>[View file]</b> zeigt wählbare Felder der gesamten Datenbank an<br>');
    childWindow.document.write('    und ermöglicht deren statistische Auswertung.<br>');
    childWindow.document.write('<b>[Close file]</b> schließt die aktuelle Datenbank.<br>');
    childWindow.document.write('<br>');
    childWindow.document.write('Datenbank-Beispiel mit Header und drei nachfolgenden Datensätzen:<br>');
    childWindow.document.write('    Index;Name;Vorname;Sex;Telefon#<br>');
    childWindow.document.write('    1;Zenz;Eva;w;01/7234219#<br>');
    childWindow.document.write('    2;Meier;Fritz;m;01/6157730#<br>');
    childWindow.document.write('    3;Adam;Franz;m;01/2315643<br>');
    childWindow.document.write('Eine solche Textdatei kann mit jedem einfachen Editor erzeugt werden,<br>');
    childWindow.document.write('aber auch im internen Editor (mit <b>[Editor on/off]</b> und <b>[Create file]</b>).<br>');
    childWindow.document.write('<br>');
    childWindow.document.write('Befinden sich im aktuellen Ordner JPG-Bilder mit den Dateinamen<br>');
    childWindow.document.write('"Feld1_Feld2.jpg", dann werden die Bilder bei den entsprechenden<br>');
    childWindow.document.write('Datensätzen automatisch angezeigt (beispielsweise "Zenz_Eva.jpg").<br>');
    childWindow.document.write('<br>');
    childWindow.document.write('Jede im Hauptspeicher erzeugte Datenbank kann mit <b>[Save file]</b> nur<br>');
    childWindow.document.write('im Download-Ordner abgespeichert werden (wegen Internet-Sicherheit).<br>');
    childWindow.document.write('Sie kann dann - falls gewünscht - im jeweiligen Betriebssystem vom<br>');
    childWindow.document.write('Download-Ordner in jeden anderen Ordner kopiert werden.<br>');
    childWindow.document.write('<br>');
    childWindow.document.write('<b>Ein Anwendungsbeispiel:</b><br>');
    childWindow.document.write('(1) Laden der Demo-Datenbank "demofile.txt" mit [Demo].<br>');
    childWindow.document.write('(2) Anhängen eines neuen Datenfeldes "Hobby" mit [WriteTo Editor]<br>');
    childWindow.document.write('    und [Insert field] und [Create file]. Dann Eingaben in "Hobby".<br>');
    childWindow.document.write('(3) Erster Datenfilter "Ort = Wien" mit [Filter on].<br>');
    childWindow.document.write('(4) Zweiter Datenfilter "Sex = w" mit [Filter on].<br>');
    childWindow.document.write('(5) Sortieren nach dem Datenfeld "Groesse" mit [Sort file].<br>');
    childWindow.document.write('(6) Listenausgabe von "Name", "Sex", "Ort", "Groesse" mit [View file].<br>');
    childWindow.document.write('    und statistische Auswertung des Datenfeldes "Groesse".<br>');
    childWindow.document.write('(7) Datenbank speichern als "demofile1.txt" mit [Save file].<br>');
    childWindow.document.write('<br>');
    childWindow.document.write('</body></html>');
   }


// Variable für die Datenfelder
var header = "";        // String der Datenfeld-Namen
var fsep   = ";";       // Datenfeld-Trennzeichen
var fname  = [];        // Datenfeld-Namen als Array
var fmax   = 0;         // Anzahl der Datenfelder
var fsize  = "15px";    // Datenfeld-Schriftgröße

// Bildkoordinaten der Datenfelder festlegen
var x1 = 30;            // x-Position des ersten Feldnamens
var x2 = x1 + 120;      // x-Position des ersten Datenfeldes
var y1 = 250;           // y-Position des ersten Feldnamens
var y2 = 250;           // y-Position des ersten Datenfeldes
var hy = 26;            // vertikaler Abstand der Datenfelder
var wx = 45;            // horizontale Breite der Datenfelder

// Variable für die Datensätze
var sep = '#';          // Datensatz-Tennzeichen
var file0;              // Datenbank-Datei als String
var allrec = [];        // Datensätze als Array
var orgrec = [];        // erste Kopie von allrec
var coprec = [];        // zweite Kopie von allrec
var rmax   = 0;         // Anzahl der Datensätze
var rnum   = 0;         // Laufende Datensatz-Nummer (Index)
var filt   = false;     // Steuervariable für Datenfilterung
var imageName = '';     // Name einer zum Datensatz passenden Bilddatei
var komma  = ',';       // interne Trennvariable

// Steuervariable
var fileLoad  = false;  // Kennzeichen für geladene Datenbank
var readFirst = false;  // Kennzeichen für eingelesenen Datensatz
var writeNew  = false;  // Kennzeichen für neuen Datensatz
var txtvisi   = false;  // Kennzeichen für die Sichtbarkeit des Texteditors
var transfer  = false;  // Kennzeichen für Datentransfer in den Texteditor

// Erstes und zweites Index-Array zur Datenbank-Sortierung
var index0 = [];        // originales Indexarray
var index1 = [];        // beliebiges Datenarray

// Kindfenster zur Listendarstellung und zum Listenausdruck
var childWindow;

// Minimale Datenbank
var file1 = 'Index;Name;Vorname;Sex;Telefon#' + '\n' +
            '1;Zenz;Eva;w;01/82765409#'+ '\n' +
            '2;Meier;Fritz;m;01/23408641#' + '\n' +
            '3;Adam;Franz;m;01/5376627';
```

```
function createStruc() {
// Feldstruktur erzeugen
  for (i = 0; i < fmax; i++) {
    s1 = '['+i+ ']  ' + fname[i]+' ';
    s2 = "feld"+i;  // Feldnamen-ID
    var f = document.createElement("TEXT");
    var t = document.createTextNode(s1);
    f.appendChild(t);
    f.setAttribute("id",s2);
    document.body.appendChild(f);
    document.getElementById(s2).style.fontStyle = "italic";
    document.getElementById(s2).style.fontSize = fsize;
    setPosition(s2,x1,y1+i*hy);
  }
}

function removeStruc() {
// Feldstruktur entfernen
  for (i = 0; i < fmax; i++) {
    s1 = "feld"+i;  // Feldnamen-ID
    s2 = "data"+i;  // Felddaten-ID
    var node1 = document.getElementById(s1);
    var node2 = document.getElementById(s2);
    node1.parentNode.removeChild(node1);
    node2.parentNode.removeChild(node2);
  }
}

// Entfernt führende und nachfolgende Leerzeichen eines Strings
function myTrim(x) { return x.replace(/^\s+|\s+$/gm,'') }

function setPosition(element,x,y) {
// absolute Positionierung
   var el = document.getElementById(element);
   el.style.position = "absolute";
   el.style.left = x + "px";
   el.style.top =  y + "px";
}

function copy0() {
// allrec -> orgrec: originale Datenbank sichern
   orgrec.length = 0;
   var n = allrec.length;
   for (j = 0; j < n; j++) { orgrec.push(allrec[j]); }
}

function copy1() {
// allrec -> coprec: Datenbank zum Filtern kopieren
   coprec.length = 0;
   var n = allrec.length;
   for (j = 0; j < n; j++) { coprec.push(allrec[j]); }
   allrec.length = 0;
}

function copy2() {
// orgrec -> allrec: gesicherte, originale Datenbank wieder erstellen
   allrec.length = 0;
   var n = orgrec.length;
   for (j = 0; j < n; j++) { allrec.push(orgrec[j]); }
   rmax = n;
}

function filterOff() {
// gesetzte Filter ausschalten
   if (!readFirst) { alert('Zuerst mit [Read record] einen Datensatz einlesen!'); return; }
   copy2();
   rnum = -1;
   nextRec();
   filt = false;
   alert('Datenfilter aufgehoben !');
}

function filterRec() {
// Datensätze mehrstufig filtern, mit Feldwerten <, =, >, ? Suchbegriff
  if (!readFirst) { alert('Zuerst mit [Read record] einen Datensatz einlesen!'); return; }
  if ( !filt ) { copy0(); }
  copy1();
  var gesucht = ' x , ? , y ';
  var info = ' Feldnummer x, Operation (<,=,>,?), Suchbegriff y ';
  gesucht = prompt(info,gesucht);
  gesucht = myTrim(gesucht);
  var s = gesucht.split(',');
  var a = myTrim(s[0]);
```

```
  if ( isNaN(a) || (a < 1) || (a > fmax-1) ) {
    alert('Falsche Datenfeldnummer!');
    filterOff();
    return;
  }
  var b = myTrim(s[2]);
  var c = myTrim(s[1]);
  var okay = false;
  if ( (c == '<') || (c == '=') || (c == '>') || (c == '?') ) { okay = true; }
  if ( !okay) {
    alert('Falsche Vergleichsoperation!');
    filterOff();
    return;
  }
  if (c == '?') {
     b = b.toUpperCase();
     var count = 0;
     for (j = 0; j < rmax; j++) {
       var rec = coprec[j].split(fsep);
       s = rec[a].toUpperCase();
       if ( s.includes(b) ) { count++; allrec.push(coprec[j]); }
     }
  }
  if (c == '<') {
     var count = 0;
     for (j = 0; j < rmax; j++) {
       var rec = coprec[j].split(fsep);
       s = rec[a];
       if ( isNaN(s) ) {
          if (s < b) { count++; allrec.push(coprec[j]); }
       }
       else {
          if (1*s < 1*b) { count++; allrec.push(coprec[j]); }
       }
     }
  }
  if (c == '=') {
     var count = 0;
     for (j = 0; j < rmax; j++) {
       var rec = coprec[j].split(fsep);
       s = rec[a];
       if (s == b) { count++; allrec.push(coprec[j]); }
     }
  }
  if (c == '>') {
     var count = 0;
     for (j = 0; j < rmax; j++) {
       var rec = coprec[j].split(fsep);
       s = rec[a];
       if ( isNaN(s) ) {
          if (s > b) { count++; allrec.push(coprec[j]); }
       }
       else {
          if (1*s > 1*b) { count++; allrec.push(coprec[j]); }
       }
     }
  }
  rmax = count;
  alert(count + ' Datensätze gefunden !');
  if (count == 0) { filterOff(); return; }
  rnum = -1;
  nextRec();
  filt = true;
}

function sortFile() {
// Gesamte Datenbank nach einem Datenfeld sortieren
  if (!readFirst) { alert('Zuerst mit [Read record] einen Datensatz einlesen!'); return; }
  var info = 'Datenfeldnummer (x) eingeben';
  var n = prompt(info,'x');
  var a = myTrim(n);
  if ( isNaN(a) || (a < 0) || (a > fmax-1) ) {
    alert('Falsche Datenfeldnummer!');
    return;
  }
  for (j = 0; j < rmax; j++) {
    var rec = allrec[j].split(fsep);
    index0[j] = rec[0];
    index1[j] = rec[a];
  }
```

```
  if ( isNaN(rec[a]) ) {
    for (i = 0; i < rmax; i++) {
     for (j = i+1; j < rmax; j++) {
      if (index1[j] < index1[i]) {
        x = index1[i]; index1[i] = index1[j]; index1[j] = x;
        y = index0[i]; index0[i] = index0[j]; index0[j] = y;
        z = allrec[i]; allrec[i] = allrec[j]; allrec[j] = z;
      }
     }
    }
  }
  if ( !isNaN(rec[a]) ) {
    for (i = 0; i < rmax; i++) {
     for (j = i+1; j < rmax; j++) {
      if (1*index1[j] < 1*index1[i]) {
        x = index1[i]; index1[i] = index1[j]; index1[j] = x;
        y = index0[i]; index0[i] = index0[j]; index0[j] = y;
        z = allrec[i]; allrec[i] = allrec[j]; allrec[j] = z;
      }
     }
    }
  }
  alert('Datenbank sortiert!');
  rnum = - 1;
  nextRec();
}

function readRec() {
// Einen aktuellen Datensatz entsprechend der Indexnummer auswählen
  if ( !fileLoad ) { alert('Zuerst Datenbank erzeugen!'); return; }
  document.getElementById('pic').style.visibility = "hidden";
  var info = '( 1 <= n <= ' + rmax + ' )';
  var num = prompt('Datensatz-Nummer ' + info, 1);
  num = myTrim(num);
  if ( isNaN(num) || (num < 1) || (num > rmax)) { num = 1; }
  rnum = num - 1;
  var rec = allrec[rnum].split(fsep);
  var s1,s2;
  if (!readFirst) {
    for (i = 0; i < fmax; i++) {
      s1 = rec[i];
      s2 = "data"+i;
      var f = document.createElement("INPUT");
      f.setAttribute("id",s2);
      document.body.appendChild(f);
      document.getElementById(s2).style.fontSize = fsize;
      document.getElementById(s2).size = wx;
      document.getElementById(s2).value = s1;
      setPosition(s2,x2,y2+i*hy);
      readFirst = true;
    }
    document.getElementById("data0").readOnly = true;
    document.getElementById("data0").style.backgroundColor = "AntiqueWhite";
    document.getElementById("data0").style.color= "red";
    imageName = rec[1] + '_' + rec[2];
    loadImage(imageName);
  }
  else {
    for (i = 0; i < fmax; i++) {
      s1 = rec[i];
      s2 = "data"+i;
      document.getElementById(s2).value = s1;
    }
    imageName = rec[1] + '_' + rec[2];
    loadImage(imageName);
  }
  document.getElementById("feld1").focus();
  transfer = false;
}

function nextRec() {
// Zum nachfolgenden Datensatz gehen
  if (!readFirst) { alert('Zuerst mit [Read record] einen Datensatz einlesen!'); return; }
  if (rnum < rmax - 1) { rnum = rnum + 1 }
  var rec = allrec[rnum].split(fsep);
  var s1,s2;
  for (i = 0; i < fmax; i++) {
    s1 = rec[i];
    s2 = "data"+i;
    document.getElementById(s2).value = s1;
  }
```

```
  imageName = rec[1] + '_' + rec[2];
  loadImage(imageName);
  document.getElementById("feld1").focus();
}

function lastRec() {
// Zum vorangehenden Datensatz gehen
  if (!readFirst) { alert('Zuerst mit [Read record] einen Datensatz einlesen!'); return; }
  if (rnum > 0) { rnum = rnum - 1 }
  var rec = allrec[rnum].split(fsep);
  var s1,s2;
  for (i = 0; i < fmax; i++) {
    s1 = rec[i];
    s2 = "data"+i;
    document.getElementById(s2).value = s1;
  }
  imageName = rec[1] + '_' + rec[2];
  loadImage(imageName);
  document.getElementById("feld1").focus();
}

function newRec() {
// Einen neuen Datensatz anlegen
  if (!readFirst) { alert('Zuerst mit [Read record] einen Datensatz einlesen!'); return; }
  if ( filt ) { alert('Zuerst den Datenfilter aufheben!'); return; }
  var s1,s2;
  for (i = 0; i < fmax; i++) {
    s1 = '';
    if (i == 0) { s1 = (1)*rmax + 1; }
    s2 = "data"+i;
    document.getElementById(s2).value = s1;
  }
  writeNew = true;
  imageName = '';
  loadImage(imageName);
  document.getElementById("data1").focus();
}

function writeRec() {
// Einen neuen oder geänderten Datensatz speichern
  if (!readFirst) { alert('Zuerst mit [Read record] einen Datensatz einlesen!'); return; }
  if ( filt ) { alert('Zuerst den Datenfilter aufheben!'); return; }
  var rec = '';
  var s1,s2,s3;
  for (i = 0; i < fmax; i++) {
    s2 = "data"+i;
    s1 = document.getElementById(s2).value;
    if ((writeNew) && (i == 0))  { s1 = (1)*rmax + 1; s3 = s1; }
    if ((!writeNew) && (i == 0)) { s1 = (1)*rnum + 1; s3 = s1; }
    rec = rec + s1 + ';';
  }
  document.getElementById("data0").value = s3;
  if (writeNew) {
  // Datenbank neu indizieren
    allrec.push(rec);
    rmax = rmax + 1;
    rnum = rmax;
    allrec.splice(rnum,1);
    var s1 = header + sep + '\n';
    for (i = 0; i < rmax; i++) {
      t1 = allrec[i];
      p  = t1.indexOf(fsep);
      t2 = t1.substr(p);
      t1 = (1*i+1) + t2;
      allrec[i] = t1;
      s1 = s1 + allrec[i] + sep;
    }
    file0 = s1.substr(0,s1.length-1);
    alert('Neuer Record gespeichert !');
  }
  else {
    allrec[rnum] = rec;
    rmax = allrec.length;
    s1 = header + sep + '\n';
    for (i = 0; i < rmax; i++) { s1 = s1 + allrec[i] + sep; }
    file0 = s1.substr(0,s1.length-1);
    alert('Aktueller Record geändert !');
  }
  writeNew = false;
}
```

```
function delRec() {
// Den aktuellen Datensatz löschen
  if (!readFirst) { alert('Zuerst mit [Read record] einen Datensatz einlesen!'); return; }
  if ( filt ) { alert('Zuerst den Datenfilter aufheben!'); return; }
  var ein = confirm('Aktuellen Datensatz wirklich löschen?');
  if (!ein) { return; }
  if (rmax == 1) { alert('Eine weitere Löschung ist nicht mehr möglich !'); return; }
  // Datenbank neu indizieren
  rmax = rmax - 1;
  allrec.splice(rnum,1);
  var s1 = header + sep + '\n';
  for (i = 0; i < rmax; i++) {
    t1 = allrec[i];
    p  = t1.indexOf(fsep);
    t2 = t1.substr(p);
    t1 = (1*i+1) + t2;
    allrec[i] = t1;
    s1 = s1 + allrec[i] + sep;
  }
  file0 = s1.substr(0,s1.length-1);
  rnum = rnum - 1;
  alert('Datensatz gelöscht und Datenbank neu indiziert!');
  nextRec();
}

function printRec() {
// Den aktuellen Datensatz drucken
  if (!readFirst) { alert('Zuerst mit [Read record] einen Datensatz einlesen!'); return; }
  document.getElementById('btns').style.visibility = "hidden";
  print();
  document.getElementById('btns').style.visibility = "visible";
}

function seekRec() {
// Einen Begriff in einem Datenfeld suchen (caseinsensitiv und nur am Feldanfang)
  if (!readFirst) { alert('Zuerst mit [Read record] einen Datensatz einlesen!'); return; }
  var found = false;
  var count = 0;
  var info = 'Datenfeldnummer (x) und Suchbegriff (y) eingeben';
  var gesucht = ' x , y ';
  gesucht = prompt(info,gesucht);
  gesucht = myTrim(gesucht);

  var s = gesucht.split(',');
  var a = myTrim(s[0]);
  var b = myTrim(s[1]);
  b = b.toUpperCase();
  var len = b.length;
  if ( isNaN(a) || (a < 1) || (a > fmax-1) ) {
    alert('Falsche Datenfeldnummer!');
    return;
  }
  for (j = 0; j < rmax; j++) {
    rec = allrec[j].split(fsep);
    index0[j] = rec[0];
    index1[j] = rec[a];
    c = index1[j].toUpperCase();
    c = c.substring(0,len);
    if (c == b) {
      found = true;
      count++;
      posi = j;
      rec = allrec[posi].split(fsep);
      for (i = 0; i < fmax; i++) {
        s1 = rec[i];
        s2 = "data"+i;
        document.getElementById(s2).value = s1;
      }
      imageName = rec[1] + '_' + rec[2];
      loadImage(imageName);
      document.getElementById("data1").focus();
      result = confirm('Suche fortsetzen ?');
      if ( !result ) { break; }
    }
  }
  if ( !found ) {
    alert('Begriff <' + s[1] + '> am Anfang von Feld <' + s[0] + '> NICHT gefunden!');
  }
  else {
    alert('Begriff <' + s[1] + '> am Anfang von Feld <' + s[0] + '> bisher ' + count +
          '-Mal gefunden!');
    rnum = posi;
  }
}
```

```
function loadFile() {
// Eine neue Datenbank öffnen
  if ( txtvisi ) { alert('Zuerst mit [Editor on/off] den Texteditor schließen!'); return; }
  if ( fileLoad ) { alert('Zuerst Datenbank mit [Close file] schließen!'); return; }
  var fileToLoad = document.getElementById("fileToLoad").files[0];
  textType = /text.*/;
  if ( !fileToLoad.type.match(textType) ) { alert('Falscher Datentyp'); return; }
  var reader = new FileReader();
  reader.readAsText(fileToLoad,"ISO-8859-1");
  reader.onload = function(event) {
     var textFile = event.target.result;
     file0  = textFile;
     allrec.length = 0;
     fname.length = 0;
     allrec = file0.split(sep);
     header = allrec.shift();
     rmax   = allrec.length;
     fname  = header.split(fsep);
     fmax   = fname.length;
     createStruc();
     copy0();
     alert('Zuerst mit [Read record] einen Datensatz einlesen!');
     document.getElementById('fileName').value = 'myfile.txt';
     document.getElementById("btn1").focus();
     transfer = false;
     fileLoad = true;
  }
}

function saveFile() {
// Die aktuelle Datenbank speichern
  if (!fileLoad) { alert('Zuerst Datenbank erzeugen!'); return; }
  if ( filt ) { alert('Zuerst den Datenfilter aufheben!'); return; }
  document.getElementById("fileToLoad").value = '';
  var fileToWrite = file0;
  var textBlob = new Blob([fileToWrite], {type:'text/plain'});
  var name = document.getElementById("fileName").value;
  var link = document.createElement("a");
  link.download = name;
  link.innerHTML = "Download File";
  link.href = window.URL.createObjectURL(textBlob);
  link.style.display = "none";
  document.body.appendChild(link);
  link.click();
  document.body.removeChild(link);
}

function imgExists(url) {
// Prüft die Existenz eines zum "Feld1_Feld2" passenden JPEG-Bildes
  var image = new Image();
  var exists = true;
  image.src = url;
  if (!image.complete) { exists = false; }
  if (image.width === 0) { exists = false; }
  return exists;
}

function loadImage(nam) {
// Zeigt ein zum "Feld1_Feld2" passendes JPEG-Bild
  nam = nam + '.jpg';
  var imgbox = document.getElementById("pic");
  imgbox.src = nam;
  var url = imgbox.src;
  var exists = imgExists(url);
  if (exists) { imgbox.style.visibility = "visible"; }
  else { imgbox.style.visibility = "hidden"; }
}

function demoFile() {
// Demo-Datenbank
  if ( txtvisi ) { alert('Zuerst mit [Editor on/off] den Texteditor schließen!'); return; }
  if ( fileLoad ) { alert('Zuerst Datenbank mit [Close file] schließen!'); return; }
  file0 = "Index;Name;Vorname;Sex;Ort;Geburt;Gewicht;Groesse#" +
          "1;Ronka;Lisa;w;Wien;1945;65;165#" +
          "2;Meier;Herbert;m;Graz;1940;87;190#" +
          "3;Dorfer;Maria;w;Graz;1950;83;160#" +
          "4;Stanka;Rudolf;m;Linz;1943;61;171#" +
          "5;Zenz;Eva;w;Wien;1942;49;167#" +
          "6;Wille;Heinz;m;Linz;1940;82;182#" +
          "7;Rutger;Herbert;m;Wien;1943;74;178#" +
          "8;Hauer;Friedl;m;Linz;1942;76;178#" +
          "9;Mueller;Roland;m;Wien;1941;81;185#" +
          "10;Wollner;Christa;w;Linz;1947;67;156#" +
```

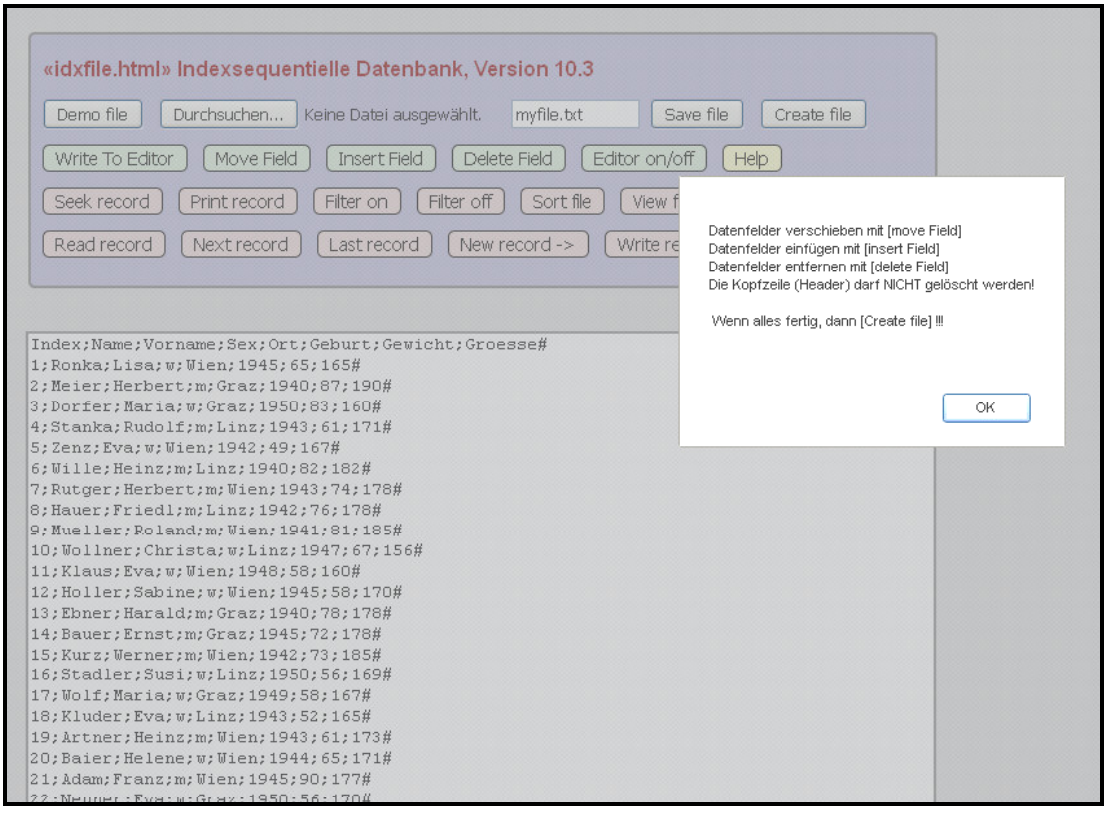
```
          "11;Klaus;Eva;w;Wien;1948;58;160#" +
          "12;Holler;Sabine;w;Wien;1945;58;170#" +
          "13;Ebner;Harald;m;Graz;1940;78;178#" +
          "14;Bauer;Ernst;m;Graz;1945;72;178#" +
          "15;Kurz;Werner;m;Wien;1942;73;185#" +
          "16;Stadler;Susi;w;Linz;1950;56;169#" +
          "17;Wolf;Maria;w;Graz;1949;58;167#" +
          "18;Kluder;Eva;w;Linz;1943;52;165#" +
          "19;Artner;Heinz;m;Wien;1943;61;173#" +
          "20;Baier;Helene;w;Wien;1944;65;171#" +
          "21;Adam;Franz;m;Wien;1945;90;177#" +
          "22;Neuner;Eva;w;Graz;1950;56;170#" +
          "23;Troll;Rudolf;m;Linz;1943;69;178#" +
          "24;Sarg;Willi;m;Wien;1950;93;182#" +
          "25;Toth;Maria;w;Wien;1947;61;157";
  allrec.length = 0;
  fname.length = 0;
  allrec = file0.split(sep);
  header = allrec.shift();
  rmax   = allrec.length;
  fname  = header.split(fsep);
  fmax   = fname.length;
  createStruc();
  copy0();
  document.getElementById('fileToLoad').value = '';
  document.getElementById('fileName').value = 'demofile.txt';
  alert('Zuerst mit [Read record] einen Datensatz einlesen!');
  document.getElementById("btn1").focus();
  ransfer = false;
  fileLoad = true;
}

function createFile() {
// Neue Datenbank erzeugen
  if ( fileLoad && !transfer ) { alert('Zuerst Datenbank mit [Close file] schließen!'); return; }
  if ( !txtvisi ) { alert('Zuerst mit [Editor on/off] den Texteditor öffnen!'); return; }
  tansfer = false;
  var text = document.getElementById('tbox').value;
  text = myTrim(text);
  var okay = true;
  if (text == '') { okay = false; }
  if (text.indexOf(sep) == -1) { okay = false; }
  if (text.indexOf(fsep) == -1) { okay = false; }
  if ( !okay ) { alert('Abbruch wegen falscher Datenbank-Struktur!'); return; }
  document.getElementById('tbox').style.visibility = "hidden";
  txtvisi = false;
  file0 = text;
  allrec.length = 0;
  fname.length = 0;
  allrec = file0.split(sep);
  header = allrec.shift();
  rmax   = allrec.length;
  fname  = header.split(fsep);
  fmax   = fname.length;
  // Datenbank neu indizieren
  var s1 = header + sep + '\n';
  for (i = 0; i < rmax; i++) {
    t1 = allrec[i];
    p  = t1.indexOf(fsep);
    t2 = t1.substr(p);
    t1 = (1*i+1) + t2;
    allrec[i] = t1;
    s1 = s1 + allrec[i] + sep;
  }
  file0 = s1.substr(0,s1.length-1);
  createStruc();
  copy0();
  alert('Zuerst mit [Read record] einen Datensatz einlesen!');
  document.getElementById('fileToLoad').value = '';
  document.getElementById('fileName').value = 'myfile.txt';
  document.getElementById("tbox").value = '';
  document.getElementById("btn1").focus();
  fileLoad = true;
}

function closeFile() {
// Datenbank schließen
  if (!fileLoad) { alert('Zuerst Datenbank erzeugen!'); return; }
  if (!readFirst) { alert('Zuerst mit [Read record] einen Datensatz einlesen!'); return; }
  if (!transfer) {
    var ein = confirm('Datenbank wirklich schließen?');
    if (!ein) { return; }
  }
```

```
  document.getElementById('fileToLoad').value = '';
  document.getElementById('fileName').value = 'myfile.txt';
  fileLoad  = false;
  readFirst = false;
  filt = false;
  if (!transfer) {
    document.getElementById('tbox').style.visibility = "hidden";
    if (!transfer) { window.location.reload(); }
  }
}

function viewFile() {
// gesamte Listendarstellung von ausgewählten Datenfeldern
  if (!readFirst) { alert('Zuerst mit [Read record] einen Datensatz einlesen!'); return; }
  if (child) { closeChild(); child = false; return; }
  var info = 'Liste der Feldnummern: 1, 2, ... ' + (fmax-1) + ', ' + '\n' + 'a = Ausgabe von ALLEN
Feldern';
  var wahl = '0,1,7,';
  wahl = prompt(info,wahl);
  if (wahl != 'a') {
     var select = 0;
     select = prompt('Statistik von Feldnummer X (0 = KEINE Statistik)',select);
     if ( isNaN(select) || (wahl.indexOf(select) == -1) ) { select = 0; }
  }
  child = true;
  childWindow = window.open('','childWindow','top=10, left=10, width=750, height=600, srollbars=yes,
menubar=no, toolbar=no');
  childWindow.document.open();
  childWindow.document.write('<html><head></head><body id="all" style="background-color:#FFFFE8;
font-family:Courier New, Arial; font-weight:bold; font-size:14px; text-size-adjust:none; margin-
left:20px; margin-bottom:40px;">');
  childWindow.document.write('<br><input type="button" value="Print" onclick="print();"><br><br>');
  var count = 0, mini = 0, maxi = 0, sum = 0, qsum = 0; mwt = 0, str = 0;
  s = '';
  for (j = 0; j < rmax; j++) {
      rec = allrec[j].split(fsep);
      t = '';
      for (i = 0; i < fmax; i++) {
        c = i + ',';
        if (wahl == 'a') { t = t + rec[i] + ';'; }
        if ((wahl != 'a') && (wahl.indexOf(c) > -1)) {
           t = t + rec[i] + ';';
           if ((select > 0) && (i == select)) {
              count++;
              sum = sum + 1*rec[i];
              qsum = qsum + 1*rec[i]*rec[i];
              if (j == 0) { mini = rec[i]; maxi = rec[i]; }
              if (rec[i] < mini) { mini = rec[i]; }
              if (rec[i] > maxi) { maxi = rec[i]; }
           }
        }
      }
      s = s + t + '<br>';
  }
  mwt  = sum / count;
  zahl = qsum/count - (mwt*mwt);
  str  = Math.sqrt(zahl);
  result = 'Feldnummer = ' + select + '<br>' +
           'Anzahl = ' + count + '<br>' +
           'Minimum = ' + mini + '<br>' +
           'Maximum = ' + maxi + '<br>' +
           'Summe = ' + sum + '<br>' +
           'Mittelwert = ' + mwt.toFixed(4) + '<br>' +
           'Streuung = ' + str.toFixed(4) + '<br>' + ' ';
  childWindow.document.write(s);
  if ((wahl != 'a') && (select > 0)) {
     childWindow.document.write('-------------------- Statistik --------------------' + '<br>');
     childWindow.document.write(result);
  }
  childWindow.document.write('</body></html>');
}

function transferData() {
// transferiert die aktuelle Datenbank in den Texteditor
  txt = file0.replace(/#/gi,'#\n');
  document.getElementById('tbox').value = txt;
}

function moveField() {
// Datenfeld verschieben
  var info = 'Ein Feld X nach Y  verschieben (von 1 bis ' + (fmax-1) + ', 0 = Abbruch)';
  var gesucht = prompt(info,'X,Y');
  var s = gesucht.split(',');
```

```
  num0 = myTrim(s[0]); if (isNaN(num0))  { alert('Abbruch: Falsche Feldnummer !'); return; }
  num1 = myTrim(s[1]); if (isNaN(num1))  { alert('Abbruch: Falsche Feldnummer !'); return; }
  if ( (num0 < 1) || (num0 > (fmax-1)) ) { alert('Abbruch: Falsche Feldnummer !'); return; }
  if ( (num1 < 1) || (num1 > (fmax-1)) ) { alert('Abbruch: Falsche Feldnummer !'); return; }
  rec = header.split(fsep);
  removed = rec.splice(num0,1);
  rec.splice(num1,0,removed);
  txt = rec + sep + '\n';
  for (j = 0; j < rmax; j++) {
    rec = allrec[j].split(fsep);
    removed = rec.splice(num0,1);
    rec.splice(num1,0,removed);
    txt = txt + rec + sep + '\n';
  }
  txt = txt.replace(/,/gi,';');
  txt = txt.replace(/\n\n/gi,'\n');
  txt = txt.substring(0,txt.length-2);
  document.getElementById('tbox').value = txt;
  file0 = txt;
  allrec = file0.split(sep);
  header = allrec.shift();
  rmax   = allrec.length;
  fname  = header.split(fsep);
  fmax   = fname.length;
  fileload = false;
}

function insertField() {
// Datenfeld anfügen
  var info = 'Neues Feld X (1 bis ' + fmax + ') einfügen (0 = Abbruch)';
  var num = prompt(info,'X');
  if ( isNaN(num) || (num < 1) || (num > fmax) ) { alert('Abbruch: Falsche Feldnummer !'); return; }
  var info = 'Neuen Feldnamen eingeben';
  var nname = prompt(info,'Feld');
  rec = header.split(fsep);
  removed = rec.splice(num,0,nname);
  txt = rec + sep + '\n';
  for (j = 0; j < rmax; j++) {
    rec = allrec[j].split(fsep);
    removed = rec.splice(num,0,'');
    txt = txt + rec + sep + '\n';
  }
  txt = txt.replace(/,/gi,';');
  txt = txt.replace(/\n\n/gi,'\n');
  txt = txt.substring(0,txt.length-2);
  document.getElementById('tbox').value = txt;
  file0 = txt;
  allrec = file0.split(sep);
  header = allrec.shift();
  rmax   = allrec.length;
  fname  = header.split(fsep);
  fmax   = fname.length;
  fileload = false;
}

function deleteField() {
// Datenfeld entfernen
  var info = 'Feld X (1 bis ' + (fmax-1) + ') entfernen (0 = Abbruch)';
  var num = prompt(info,'X');
  if ( isNaN(num) || (num < 1) || (num > (fmax-1)) ) { alert('Abbruch: Falsche Feldnummer !');
return; }
  rec = header.split(fsep);
  removed = rec.splice(num,1);
  txt = rec + sep + '\n';
  for (j = 0; j < rmax; j++) {
    rec = allrec[j].split(fsep);
    removed = rec.splice(num,1);
    txt = txt + rec + sep + '\n';
  }
  txt = txt.replace(/,/gi,';');
  txt = txt.replace(/\n\n/gi,'\n');
  txt = txt.substring(0,txt.length-2);
  document.getElementById('tbox').value = txt;
  file0 = txt;
  allrec = file0.split(sep);
  header = allrec.shift();
  rmax   = allrec.length;
  fname  = header.split(fsep);
  fmax   = fname.length;
  fileload = false;
}
```

```
function open_() {
// Editor öffnen oder schließen
  if (txtvisi) { alert('[Create file] anklicken!'); return; }
  var info = '\n' +
             ' Datenfelder verschieben mit [move Field, F4] ' + '\n' +
             ' Datenfelder einfügen mit [insert Field, F8] ' + '\n' +
             ' Datenfelder entfernen mit [delete Field, F9] ' + '\n' +
             ' Die Kopfzeile (Header) darf NICHT gelöscht werden! ' + '\n' + '\n ' +
             ' Wenn alles fertig, dann [Create file] !!!' + '\n' + ' ';
  var OK = false;
  if ( fileLoad ) { alert('Zuerst Datenbank mit [Close file] schließen!'); return; }
  txtvisi = !txtvisi;
  if (txtvisi) { alert('Neue Datenbank editieren und dann [Create file] anklicken!'); }
  document.getElementById('tbox').value = file1;
  if (txtvisi) { document.getElementById('tbox').style.visibility = "visible"; }
  else { document.getElementById('tbox').style.visibility = "hidden"; }
  if (txtvisi && transfer && okay) {
    imageName = '';
    loadImage(imageName);
    alert(info);
  }
}

function write_() {
// Datenbank in den Editor schreiben
  if (transfer) { alert('Mit [Create file] eine neue Datenbank erzeugen !'); return; }
  if (!readFirst) { alert('Zuerst mit [Read record] einen Datensatz einlesen!'); return; }
  var info = '\n' +
             ' Datenfelder verschieben mit [move Field] ' + '\n' +
             ' Datenfelder einfügen mit [insert Field] ' + '\n' +
             ' Datenfelder entfernen mit [delete Field] ' + '\n' +
             ' Die Kopfzeile (Header) darf NICHT gelöscht werden! ' + '\n' + '\n ' +
             ' Wenn alles fertig, dann [Create file] !!!' + '\n' + ' ';
  transferData();
  removeStruc();
  txtvisi = true;
  transfer = true;
  closeFile();
  okay = true;
  if (txtvisi) { document.getElementById('tbox').style.visibility = "visible"; }
  else { document.getElementById('tbox').style.visibility = "hidden"; }
  if (txtvisi && transfer && okay) {
    imageName = '';
    loadImage(imageName);
    alert(info);
  }
}

function move_() {
// Ein Datenfeld verschieben
  if (!transfer) { return; }
  if (txtvisi && transfer) { moveField(); }
  if (txtvisi) { document.getElementById('tbox').style.visibility = "visible"; }
  else { document.getElementById('tbox').style.visibility = "hidden"; }
  if (txtvisi && transfer && okay) {
    imageName = '';
    loadImage(imageName);
  }
}

function insert_() {
// Ein  Datenfeld einfügen
  if (!transfer) { return; }
  if (txtvisi && transfer) { insertField(); }
  if (txtvisi) { document.getElementById('tbox').style.visibility = "visible"; }
  else { document.getElementById('tbox').style.visibility = "hidden"; }
  if (txtvisi && transfer && okay) {
    imageName = '';
    loadImage(imageName);
  }
}

function delete_() {
// Ein  Datenfeld löschen
  if (!transfer) { return; }
  if (txtvisi && transfer) { deleteField(); }
  if (txtvisi) { document.getElementById('tbox').style.visibility = "visible"; }
  else { document.getElementById('tbox').style.visibility = "hidden"; }
  if (txtvisi && transfer && okay) {
    imageName = '';
    loadImage(imageName);
  }
}
```

```
</script>
</head>

<body oncontextmenu="return false">
<fieldset id="fs">
<div id="btns">
<h3 id="tit">«idxfile.html» Indexsequentielle Datenbank, Version 10.5</h3>
<input type="button" value="Demo file" class="bs2" onclick="demoFile()">  
<input type="file" id="fileToLoad" class="bs2" onchange="loadFile()">  
<input type="text" id="fileName" class="bs2" size="12" value="myfile.txt">
<input type="button" value="Save file" class="bs2" onclick="saveFile()">  
<input type="button" value="Create file" class="bs2" onclick="createFile()">  
<br><br>
<input type="button" value="Write To Editor" class="bs1" onclick="write_()">  
<input type="button" value="Move Field" class="bs1" onclick="move_();">  
<input type="button" value="Insert Field" class="bs1" onclick="insert_();">  
<input type="button" value="Delete Field" class="bs1" onclick="delete_();">  
<input type="button" value="Editor on/off" class="bs1" onclick="open_()">  
<input type="button" value="Help" class="bs3" onclick="help_()">
<br><br>
<input type="button" value="Seek record" class="bs" onclick="seekRec()">  
<input type="button" value="Print record" class="bs" onclick="printRec()">  
<input type="button" value="Filter on" class="bs" onclick="filterRec()">  
<input type="button" value="Filter off" class="bs" onclick="filterOff()">  
<input type="button" value="Sort file" class="bs" onclick="sortFile()">  
<input type="button" value="View file" class="bs" onclick="viewFile()">  
<input type="button" value="Close file" class="bs" onclick="closeFile()">  
<br><br>
<input type="button" id="btn1" value="Read record" class="bs" onclick="readRec()">  
<input type="button" value="Next record" class="bs" onclick="nextRec()">  
<input type="button" value="Last record" class="bs" onclick="lastRec()">  
<input type="button" value="New record ->" class="bs" onclick="newRec()">  
<input type="button" value="Write record" class="bs" onclick="writeRec()">  
<input type="button" value="Delete record" class="bs" onclick="delRec()">  
<br><br>
</div>
</fieldset>
<br><br>
<img id="pic" src="" onload="loadImage(imageName)">
<textarea id="tbox" value="" cols="85" rows="26" wrap="off" style="visibility:hidden;">
</textarea>
</body>
</html>
```



Die Grafik zeigt den internen Texteditor zur Änderung der Datenstruktur.

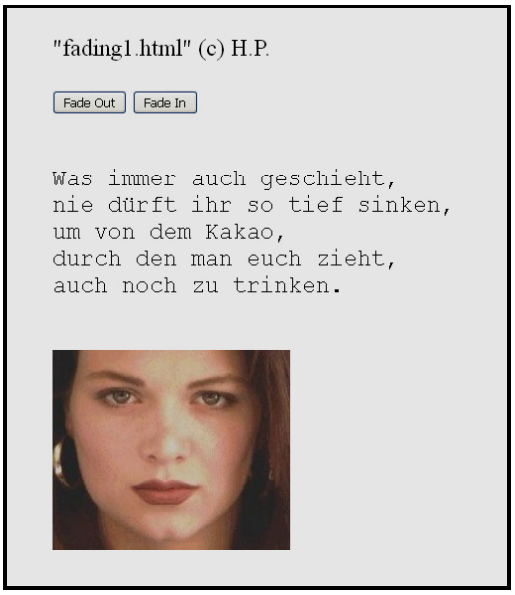
[5.3] Der Fading-Effekt

Fading bedeutet, dass die Sichtbarkeit eines HTML-Elementes schrittweise sanft verändert wird. Alle dazu notwendigen Funktionen liefert die JavaScript-Datei „**fading.js**“.

```
/* "fading.js" (c) H.P. */
function getPosition(element) {
/* liefert die absolute Position eines HTML-Elementes */
  var el = document.getElementById(element);
  var xPos = 0;
  var yPos = 0;
  while (el) {
    if ((el.tagName == "body") || (el.tagName == "BODY")) {
      var xScroll = el.scrollLeft || document.documentElement.scrollLeft;
      var yScroll = el.scrollTop || document.documentElement.scrollTop;
      xPos += (el.offsetLeft - xScroll + el.clientLeft);
      yPos += (el.offsetTop  - yScroll + el.clientTop);
    }
    else {
      xPos += (el.offsetLeft - el.scrollLeft + el.clientLeft);
      yPos += (el.offsetTop  - el.scrollTop + el.clientTop);
    }
    el = el.offsetParent;
  }
  var position = new Object();
  position.x = xPos;
  position.y = yPos;
  return position;
}

function setPosition(element,x,y) {
/* setzt die absolute Position eines HTML-Elementes */
   var el = document.getElementById(element);
   el.style.position = "absolute";
   el.style.left = x + "px";
   el.style.top =  y + "px";
}

function Opacity(element, percent) {
/* Verändert die Opacity eines HTML-Elementes in Prozenten */
   var el = document.getElementById(element);
   el.style.opacity = percent / 100;
}

function Fade(element, start, end, msecs) {
/* Fading eines HTML-Elementes */
   var el = document.getElementById(element);
   var INTVAL = 10;
   el.FADE_Start = start;
   el.FADE_Level = start;
   el.FADE_End  = end;
   var stepval = Math.abs(start - end) / (msecs / INTVAL);
   el.FADE_Step = end > start ? stepval : -stepval;

   el.FADE_Fun  = setInterval(runFade, INTVAL);

   function runFade() {
      el.FADE_Level += el.FADE_Step;
      if (el.FADE_Level >= Math.max(el.FADE_Start, el.FADE_End) ||
          el.FADE_Level <= Math.min(el.FADE_Start, el.FADE_End)) {
        el.FADE_Level = el.FADE_End;
        clearInterval(el.FADE_Fun);
      }
      Opacity(element, el.FADE_Level);
   }
}

function FadeOut(element, msecs) {
/* Outfading eines HTML-Elementes */
   Fade(element, 100, 0, msecs);
}

function FadeIn(element, msecs) {
/* Infading eines HTML-Elementes */
  Fade(element, 0, 100, msecs);
}
/* Ende von "fading.js" */
```

[5.3.1] „fading1.html“

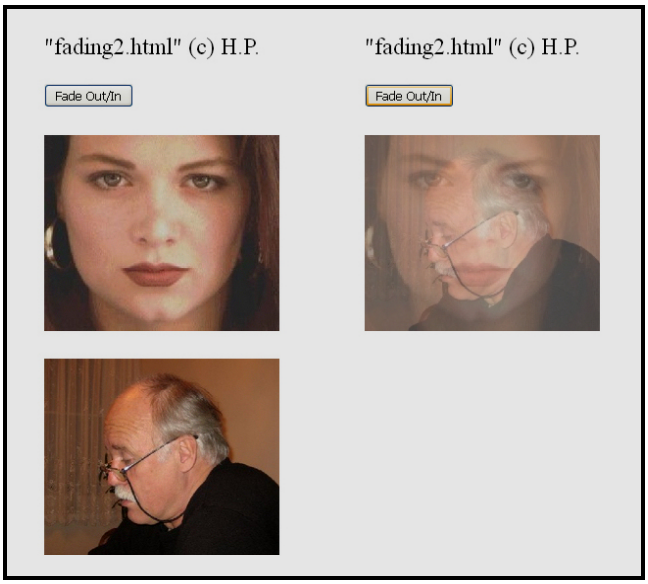
Ein Programm zum schrittweisen, sanften Ein- und Ausblenden (Fading) von Bereichen.




```
<!DOCTYPE html">
<html>
<head>
<title>Fading-Demo-1</title>
<meta charset="ISO-8859-1">
<meta name="description" content="Fading-Demo-1">
<meta name="author" content="Paukert Herbert">
<style>
  body { margin-left:100px; font-size: 24px; font-style: bold; }
</style>

<script src="fading.js"></script>

<script>
window.onload = function() {
  document.getElementById('i1').onclick = function() {
    Fade('i3',100,0,1000)
  }
  document.getElementById('i2').onclick = function() {
    Fade('i3',0,100,1000)
  }
}
</script>
</head

<body>
<br>
<p>"fading1.html" (c) H.P.</p>
<input id='i1' type='submit' value='Fade Out'>
<input id='i2' type='submit' value='Fade In'>
<br><br>
<span id='i3'>
<pre>
Was immer auch geschieht,
nie dürft ihr so tief sinken,
um von dem Kakao,
durch den man euch zieht,
auch noch zu trinken.
</pre>
<br>
<img src="frau.jpg" width="240px" alt='[ hier kann auch ein Bild sein ]'>
<br>
</span>
</body>
</html>
```

[5.3.2] „fading2.html“

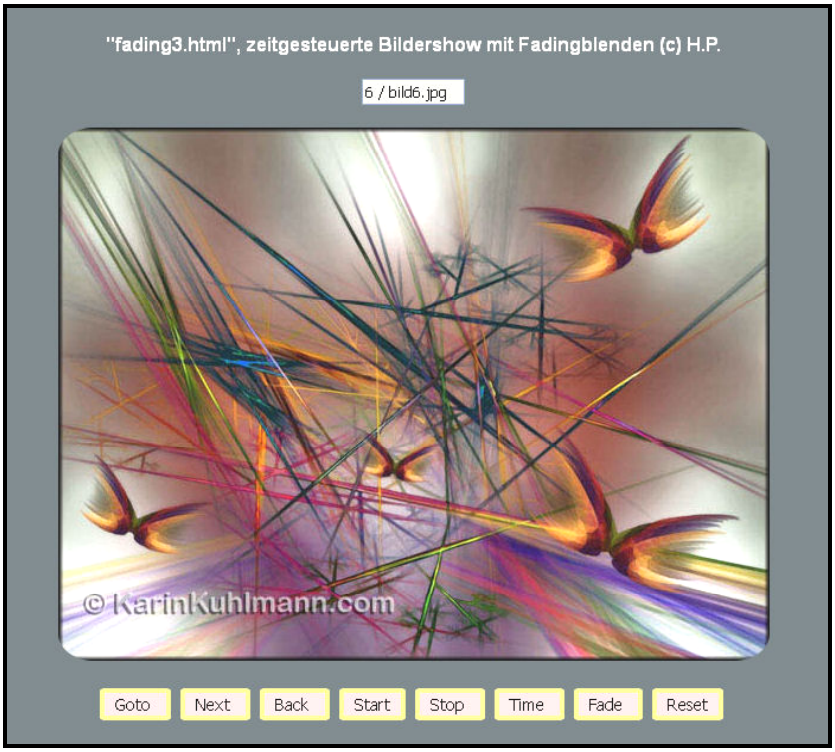
Ein Programm zum Überblenden von zwei Bildern (Fading).




```
<!DOCTYPE html">
<html>
<head>
<title>Fading-Demo-2</title>
<meta charset="ISO-8859-1">
<meta name="description" content="Fading-Demo-2">
<meta name="author" content="Paukert Herbert">

<style>
  body { margin-left:100px; font-size: 24px; font-style: bold; }
</style>

<script src="fading.js"></script>

<script>
  window.onload = function() {
    first = true;
    x0 = getPosition('i1').x;
    y0 = getPosition('i1').y;

    document.getElementById('i0').onclick = function() {
      setPosition('i2',x0,y0);
      if (first) {
         FadeOut('i1',2000);
         FadeIn('i2',2000);
      }
      else {
         FadeOut('i2',2000);
         FadeIn('i1',2000);
      }
      first = !first;
    }
  }
</script>
</head

<body>
<br>
<p>"fading2.html" (c) H.P.</p>
<input id='i0' type='submit' value='Fade Out/In'>
<br><br>
<img src="frau.jpg" id="i1" width=240 alt="[ ERSTES Bild ]" >
<br><br>
<img src="mann.jpg" id="i2" width=240 alt="[ ZWEITES Bild ]" >
<br>
</body>
</html>
```

[5.3.3] Zeitgesteuerte Bildershow (fading3.html“)

Ein Programm zur zeitgesteuerten Bildershow mit Fading-Blenden.




```
<!DOCTYPE html>
<head>
<title> Fading-Demo-3 </title>
<meta charset="ISO-8859-1">
<meta name="description" content="Fading-Demo-3">
<meta name="author" content="Herbert Paukert">

<style type="text/css">

   body  { background-color:#687378; color:white;
           font-family:Arial; font-size:16px; font-weight:normal; text-size-adjust: none;
           text-align: center; }
   .btn  { border: 4px solid #FFFF88; border-radius: 10%; background-color: #EED8D8;
           font-size: 14px; }
   #pic1 { height: 450px; border-radius: 5%; }
   #pic2 { height: 450px; border-radius: 5%; }

</style>

<script src="fading.js"></script>

<script>
   function myTrim(x) { return x.replace(/^\s+|\s+$/gm,'') }

   var pictA = new Image();
   var pictB = new Image();

   var num = 1;
   var max = 8;
   var name = 'bild';
   bild = new Array(max);
   for (var i = 1; i <= max; i++) { bild[i] = name + i + '.jpg'; }

   var active;
   var zeit = 3000;
   var zeit0 = 3000;
   var ShowRun = false;
   var FadeTime = 1000;
   var FadeTime0 = 1000;
   var FadeRun = false;
```

```
window.onload = function() {
// Zwei Images übereinanderlegen
  pictB.src = bild[1];
  document.pic2.src = pictB.src;
  setPosition("pic1",getPosition("pic2").x,getPosition("pic2").y);
  pictA.src = bild[1];
  document.pic1.src = pictA.src;
  document.pic1.style.visibility='hidden';
  document.formu.ausgabe.value = num + ' / ' + bild[1];
  window.scrollTo(0,0);
}

function GotoImage()  {
// Zu einem Bild springen
  if (ShowRun ) { return; }
  zahl = prompt('Nummer der Bildseite zwischen 1 und '+(max),num);;
  zahl = myTrim(zahl);
  if (!isNaN(zahl)) {
    num = zahl - 1;
    if (num <= 0) { num = 0; }
    if (num >= max) { num = max-1 ; }
  }
  else { num = 0; }
  GotoNext();
}

function GotoNext() {
// Zum nachfolgenden Bild wechseln
// oder eine Bildershow ausführen
  num = num + 1;
  if (num > max) { num = 1; }
  document.formu.ausgabe.value = num + ' / ' + bild[num];
  if (FadeRun) {
     document.pic1.style.opacity = 1;
     document.pic2.style.opacity = 0;
     setPosition("pic1",getPosition("pic2").x,getPosition("pic2").y);
     pictA.src = bild[num-1];
     document.pic1.src = pictA.src;
     FadeOut('pic1',FadeTime);
     pictB.src = bild[num];
     document.pic2.src = pictB.src;
     FadeIn('pic2',FadeTime);
   }
   else {
     pictB.src = bild[num];
     document.pic2.src = pictB.src;
   }
   if (ShowRun) {
     active = window.setTimeout("GotoNext()",zeit);
   }
}

function GotoLast() {
// Zum vorangehenden Bild wechseln
  if (ShowRun ) { return; }
  num = num - 1;
  if (num < 1) { num = max; }
  document.formu.ausgabe.value = num + ' / ' + bild[num];
  if (FadeRun) {
     document.pic1.style.opacity = 1;
     document.pic2.style.opacity = 0;
     setPosition("pic1",getPosition("pic2").x,getPosition("pic2").y);
     pictA.src = bild[num+1];
     document.pic1.src = pictA.src;
     FadeOut('pic1',FadeTime,false);
     pictB.src = bild[num];
     document.pic2.src = pictB.src;
     FadeIn('pic2',FadeTime,false);
  }
  else {
     pictB.src = bild[num];
     document.pic2.src = pictB.src;
   }
}

function ShowStart() {
// automatische Bildershow starten
  if (ShowRun) { return; }
  ShowRun = true;
  active = window.setTimeout("GotoNext()",zeit);
 }
```

```
   function ShowStop() {
   // automatische Bildershow stoppen
     ShowRun = false;
     window.clearTimeOut(active);
   }

   function GetTime() {
   // Darbietungszeit der Bilder in der Show
     if (ShowRun) { return; }
     zahl = prompt('Darbietungszeit zwischen 1000 und 9000 MSec \n ',zeit);
     zahl = myTrim(zahl);
     if (!isNaN(zahl)) {
       if ((zahl < 1000) || (zahl > 9000)) { zahl = zeit0; }
     }
     else { zahl = zeit0; }
     zeit = zahl;
   }

   function GetFade() {
   // Blendenzeit der Fadingblende
     if (FadeRun == true) {
        document.pic1.style.visibility='hidden';
        FadeRun = false;
        return;
     }
     if (FadeRun == false) {
       info = 'Blendenzeit zwischen 500 und 5000 MSec \n (kleiner als die Darbietungszeit) \n'
       zahl = prompt(info,FadeTime);
       zahl = myTrim(zahl);
       if (!isNaN(zahl)) {
         if ((zahl < 500) || ( zahl > 5000)) { zahl = FadeTime0; }
       }
       else { zahl = FadeTim0; }
       FadeTime = zahl;
       FadeRun = true;
       document.pic1.style.visibility='visible';
       document.pic2.style.visibility='visible';
       document.pic1.style.opacity = 0;
       document.pic2.style.opacity = 1;
     }
   }

   function Refresh() {
   // HTML-Seite neu laden (Reset)
      window.location.reload();
   }

</script>
</head>

<body>
<form name = "formu">
<audio id="backplay" loop autoplay><source src="backsound.mp3" controls="true"
type="audio/mp3"></audio>
<br>
<b> "fading3.html", zeitgesteuerte Bildershow mit Fadingblenden (c) H.P.</b>
<br><br>
<input type = "text" name="ausgabe" value="" size="10" readonly>
<br><br>
<img src="" id="pic1" name="pic1">
<img src="" id="pic2" name="pic2">
<br><br>
<input type = "button" class = "btn" value = " Goto  " onclick = "GotoImage()"> 
<input type = "button" class = "btn" value = " Next  " onclick = "GotoNext()"> 
<input type = "button" class = "btn" value = " Back  " onclick = "GotoLast()"> 
<input type = "button" class = "btn" value = " Start " onclick = "ShowStart()"> 
<input type = "button" class = "btn" value = " Stop  " onclick = "ShowStop()"> 
<input type = "button" class = "btn" value = " Time  " onclick = "GetTime()"> 
<input type = "button" class = "btn" value = " Fade  " onclick = "GetFade()"> 
<input type = "button" class = "btn" value = " Reset " onclick = "Refresh()"> 
</form>
</body>
</html>
```

[5.4] Drag & Drop - Methoden

Drag & Drop (Ziehen und Ablegen) sind universelle Methoden, die in JavaScript durch standardisierte Attribute, Funktionen und Events durchgeführt werden können. Dazu sind folgende Schritte notwendig:

(1) Mit dem Attribut ***draggable = 'true'*** wird ein HTML-Element ziehbar.
(***draggable = 'false'*** hingegen unterbindet das Ziehen)

(2) Mit dem Event ***ondragstart*** auf dem zu ziehenden Element wird ein Event-Handler ***drag(event)*** aufgerufen, der das Ziehen durchführt.

```
function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
}
```

(3) Mit dem Event ***ondragover*** auf dem Ablagebereich (dropzone) wird ein Event-Handler ***allowDrop(event)*** aufgerufen, der das Ablegen ermöglicht.

```
function allowDrop(ev) {
  ev.preventDefault();
}
```

(4) Mit dem Event ***ondrop*** auf dem Ablagebereich (dropzone) wird ein Event-Handler **drop(event)** aufgerufen, der das Ablegen durchführt.

```
function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
```

Die beschriebenen Events werden alle mit gehaltener und entsprechend bewegter Maus ausgeführt. Beim Drag-Vorgang wird das Ergebnis in einem ***'dataTransferObject '*** gespeichert. Dabei können verschiedene dataTransfer-Eigenschaften mit folgenden Methoden gesetzt oder gelesen werden.

- ***dataTransfer.setData(Format, Data)*** setzt die Zieldaten (***event.target.id***) und auch ihr Format (***text/plain, image/jpeg, . . .***).
- ***dataTransfer.getData(Format)*** liefert die gesetzten Daten in deren Format.
- ***dataTransfer.clearData(Format)*** entfernt gespeicherte Datenwerte.

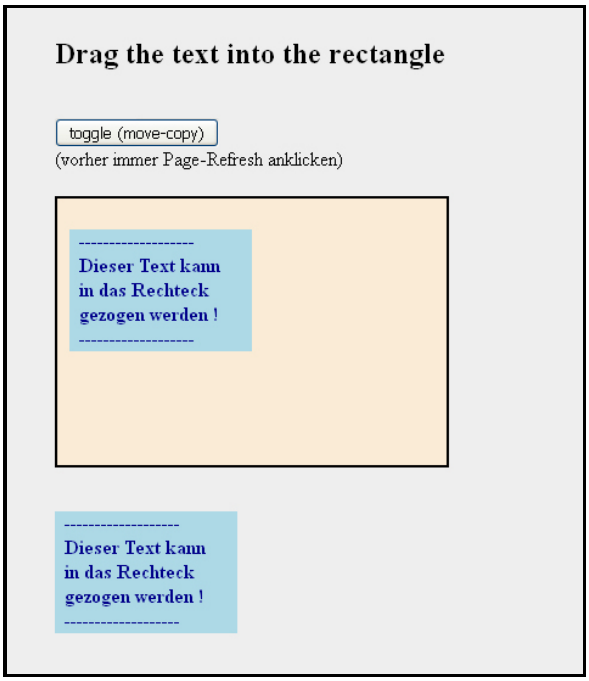
Hinweis: Wenn nach dem Drag & Drop das gezogene Element auf seiner ursprünglichen Position bestehen bleiben soll, dann entspricht das einem Kopieren des Elementes. Dann muss die Funktion **drop(event)** abgeändert werden:

```
function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data).cloneNode(true));
}
```

Dabei erzeugt die Methode ***clone.Node(true)*** eine Kopie des entsprechenden Elementes (Knotens). Diese Methode klont alle Attribute und deren Werte. Um das geklonte Element sichtbar zu machen, muss es dann mit ***appendChild()*** in das Dokument eingefügt werden.

Die drei nächsten Programme „drag01.html", „drag02.html" und „mmind.html" sind illustrative Beispiele für das beschriebene Drag & Drop.

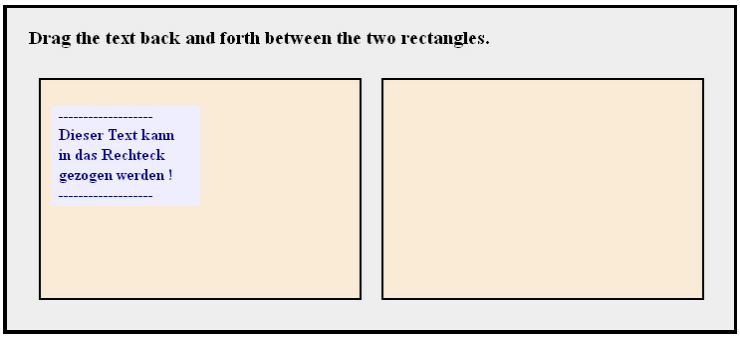
[5.4.1] „drag01.html“

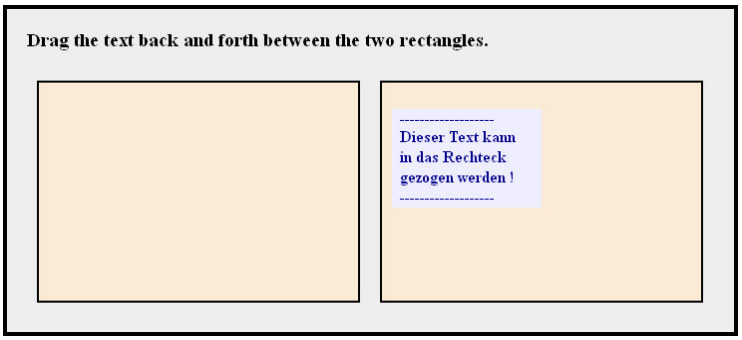



```
<!DOCTYPE html>
<html>
<head>
<title>Drag & Drop (1)</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0;" />
<meta charset="ISO-8859-1">
<meta name="author" content="H.P">

<style>
  body {
    background: #EEEEEE;
    margin: 50px;
  }
  #drop1 {
    width:  300px;
    height: 200px;
    padding: 10px;
    background: antiquewhite;
    border: 2px solid #000000;
  }
  #drag1 {
    width: 150px;
    font-weight: bold;
    color: darkblue;
    background: lightblue;
  }
</style>

<script>
  var copy = false;

  function toogle() {
    copy = !copy;
    if (copy) { alert('COPY'); }
    else { alert('MOVE'); }
  }

  function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
  }

  function allowDrop(ev) {
    ev.preventDefault();
  }
```

```
  function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    var element = document.getElementById(data);
    if (!copy) {
       ev.target.appendChild(element);
    }
    if (copy)  {
       document.getElementById("drop1").innerHTML = "";
       ev.target.appendChild(element.cloneNode(true));
    }
  }
</script>
</head>

<body>
  <h2>Drag the text into the rectangle</h2>
  <br>
  <button id="btn" onclick="toogle()"> toggle (move-copy) </button><br>
  (vorher immer Page-Refresh anklicken)<br>
  <br>

  <div id="drop1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
  <br>

  <p id="drag1" draggable="true" ondragstart="drag(event)">
    ------------------<br>
    Dieser Text kann <br>
    in das Rechteck  <br>
    gezogen werden ! <br>
    ------------------<br>
  </p>

  <!--
  Ein Image als alternativer Dragbereich:
  <br>
  <img id="drag2" src="bild.jpg" draggable="true" ondragstart="drag(event)" height="100"
   alt="Bild">
  <br>
  -->
</body>
</html>
```

[5.4.2] „drag02.html“

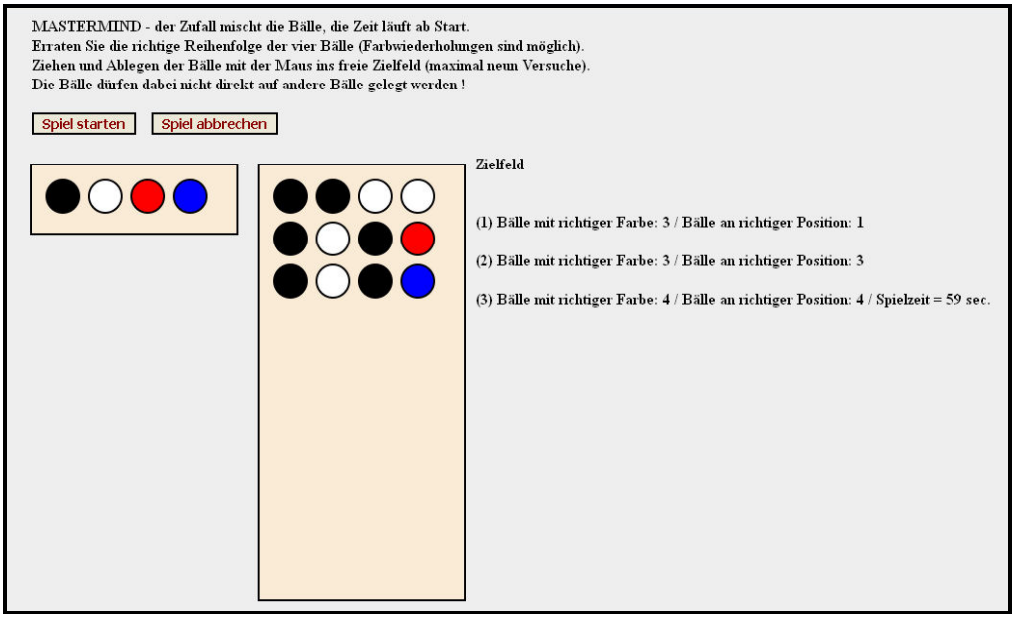




```
<!DOCTYPE html>
<html>
<head>
<title>Drag & Drop (2)</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0;" />
<meta charset="ISO-8859-1">
<meta name="author" content="H.P">

<style>
  body {
    background: #eeeeee;
    margin: 50px;
  }

  #drop1, #drop2 {
    float: left;
    width:   300px;
    height:  200px;
    margin:  10px;
    padding: 10px;
    border:  2px solid black;
    background: antiquewhite;
  }

  #drag1 {
    width: 150px;
    color: darkblue;
    font-weight: bold;
    background: #eeeeff;
  }
</style>

<script>
  function allowDrop(ev) {
    ev.preventDefault();
  }

  function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
  }

  function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
  }
</script>
</head>

<body>
  <h3>Drag the text back and forth between the two rectangles.</h3>

  <div id="drop1" ondrop="drop(event)" ondragover="allowDrop(event)">
    <p id="drag1" draggable="true" ondragstart="drag(event)" >
         -------------------<br>
         Dieser Text kann <br>
         in das Rechteck  <br>
         gezogen werden ! <br>
         -------------------<br>
    </p>
    <!--
       Ein Image als alternativer Dragbereich:
       <br>
       <img id="drag2" src="bild.jpg" draggable="true" ondragstart="drag(event)"
        height="100" alt="Bild">
       <br>
    -->
  </div>

  <div id="drop2" ondrop="drop(event)" ondragover="allowDrop(event)">
  </div>

</body>
</html>
```

[5.4.3] Ein Mastermind-Spiel (mmind.html)

Hinweis: Das vorliegende Programm ist ein **Mastermind-Spiel**. Der Zufall erzeugt vier Bälle in den Farben schwarz, weiß, rot und blau. Dabei können sich die Farben wiederholen. Mittels Drag und Drop soll der Spieler die Reihenfolge der Bälle erraten. Nach jeder Eingabe von jeweils 4 Bällen werden die Anzahlen der Positionstreffer und der Farbtreffer angezeigt. Daraus kann der Spieler die richtige Reihenfolge erschließen. Es sind maximal 9 Versuche möglich, danach erfolgt ein Spielabbruch. Zusätzlich wird noch die Spielzeit gemessen.




```
<!DOCTYPE html>
<html>
<head>
<title>Mastermind</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta charset="ISO-8859-1">
<meta name="author" content="Herbert Paukert">

<style>
  body  { background: #eeeeee;  margin-left: 50px;  font-weight: bold; }
  .btn  { border: 2px solid #000000; color: darkred; font-size: 14px; font-weight: bold; }
  #div1 {
    float:   left;
    width:   12em;
    height:  50px;
    margin:  10px;
    padding: 10px;
    border:  2px solid #000000;
    background: antiquewhite;
  }
  #div2 {
    float:   left;
    width:   12em;
    height:  430px;
    margin:  10px;
    padding: 10px;
    border:  2px solid #000000;
    background: antiquewhite;
  }

  p {
    float:  left;
    margin: 4px;
    width:  2em;
    height: 2em;
    border: 2px solid #000000;
    border-radius:50%;
  }
```

```
  #black { background: black; }
  #white { background: white; }
  #red   { background: red; }
  #blue  { background: blue; }
</style>

<script>
// globale Spielparameter
  var run = false; // Spielerlaubnis
  var cmax  = 4;   // Anzahl der Spielfiguren
  var count = 0;   // Anzahl von Drag & Drop
  var crest = 0;   // Hilfsvariable count % cmax
  var cset  = 0;   // Anzahl der Versuche (4 mal Drag & Drop)
  var farbe = 0;   // Anzahl der Farbtreffer
  var posi  = 0;   // Anzahl der Positionstreffer
  var limit = 9;   // maximal mögliche Versuche
  var vorgabe = 'black,white,red,blue,';
  var eingabe = '';
  var vor =  [];
  var ein =  [];
  var ausgabe = '';
  var starttime = 0;
  var endtime   = 0;
  var finaltime = 0;

  function startclock() {var today = new Date(); starttime = today.getTime(); }
  function endclock()   {var today = new Date(); endtime = today.getTime(); }
  function calctime()   {var time = Math.round((endtime - starttime) / 1000);
                         return time; }
  function zufall() {
  // Zufallserzeugung der Spielparameter
    vorgabe = '';
    for (i = 1; i <= 4; i++) {
      z = Math.floor(4 * Math.random()) + 1;
      if ( z == 1 ) { s = 'black'; }
      if ( z == 2 ) { s = 'white'; }
      if ( z == 3 ) { s = 'red';   }
      if ( z == 4 ) { s = 'blue';  }
      vorgabe = vorgabe + s + ','
    }
  }

  function init() {
  // Spielparameter initialisieren
    document.getElementById("div2").innerHTML = '';
    document.getElementById("div3").innerHTML = '';
    count = 0;
    crest = 0;
    cset  = 0;
    farbe = 0;
    posi  = 0;
    starttime = 0;
    endtime   = 0;
    finaltime = 0;
    run = true;
    startclock();
    zufall();
  }

  function test() {
  // Treffer überprüfen (Position und Farbe)
    ein  = eingabe.split(',');
    vor  = vorgabe.split(',');
    len  = vor.length;
    posi  = 0;
    for (i = 0; i <= len-2; i++) {
      if ( ein[i] == vor[i] ) {
         posi = posi + 1;
      }
    }
    farbe = 0;
    for (i = 0; i <= len-2; i++) {
        a = ein[i];
        for (j = 0; j <= len-2; j++) {
          if (vor[j] == a) {
             farbe = farbe + 1;
             vor[j] = '*';
             break;
          }
        }
    }
```

```
    ausgabe = 'Bälle mit richtiger Farbe: ' + farbe + '  /   ' +
              'Bälle an richtiger Position: ' + posi;
    if ( eingabe == vorgabe ) {
       run = false;
       endclock();
       finaltime = calctime();
       txt = 'Alles richtig ! Spielzeit = ' + finaltime + ' sec.'
       alert(txt);
    }
  }

  function drag(ev)
  // Ziehen eines Objektes
  {
    if (!run ) { return; }
    ev.dataTransfer.setData('text', ev.target.id);
  }

  function allowDrop(ev)
  // Ablegen des Objektes zulassen
  {
    if (!run ) { return; }
    ev.preventDefault();
  }

  function drop(ev)
  // Ablegen (Kopieren) eines Objektes
  {
    if (!run ) { return; }

    ev.preventDefault();
    var data = ev.dataTransfer.getData('text');
    var element = document.getElementById(data);
    ev.target.appendChild(element.cloneNode(true));

    var info = document.getElementById("div3").innerHTML;
    if (crest == 0) {
       cset = cset + 1;
       txt = '<br><br>' + '(' + cset + ') ';
       eingabe = '';
    }
    else { txt = ' ' }

    count = count + 1;
    crest = (count % cmax);
    info  = info + txt;
    document.getElementById("div3").innerHTML = info;
    eingabe = eingabe + data + ',';

    if (crest == 0) {
       test();
       erg = ' ';
       if (!run) { erg = ' / Spielzeit = ' + finaltime + ' sec.' }
       document.getElementById("div3").innerHTML = info + ' ' + ausgabe + erg;
    }

    if (cset == (limit + 1)) { abbruch(); }
  }

  function abbruch()
  // Spielabbruch nach 10 Versuchen
  {
    if ( !run ) { return; }
    run = false;
    endclock();
    finaltime = calctime();
    txt = 'Spielabbruch, Lösung = [ ' + vorgabe + '] Zeit = ' + finaltime + ' sec.';
    alert(txt);
    document.getElementById("div3").innerHTML = txt;
  }

</script>
</head>
```

```
<body>
     MASTERMIND - der Zufall mischt die Bälle, die Zeit läuft ab Start.<br>
     Erraten Sie die richtige Reihenfolge der vier Bälle (Farbwiederholungen
               sind möglich).<br>
     Ziehen und Ablegen der Bälle mit der Maus ins freie Zielfeld (maximal
               neun Versuche).<br>
     Die Bälle dürfen dabei nicht direkt auf andere Bälle gelegt werden !<br>
  <br>
     <button class="btn" onclick="init();"> Spiel starten </button>
     <button class="btn" onclick="abbruch();"> Spiel abbrechen </button>
  <br><br>
  Zielfeld
  <div id="div1">
    <p id="black" draggable="true" ondragstart="drag(event)"></p>
    <p id="white" draggable="true" ondragstart="drag(event)"></p>
    <p id="red"   draggable="true" ondragstart="drag(event)"></p>
    <p id="blue"  draggable="true" ondragstart="drag(event)"></p>
  </div>

  <div id="div2" draggable="false" ondragover="allowDrop(event)" ondrop="drop(event)">
  </div>
  <br>
  <div id="div3">
  <div>
</body>
</html>
```

Programmieren mit JavaScript, Teil 6
Animationen und Spiele

[6.1] Das Animations-System

Eine Vielzahl von **CSS-Anweisungen** dient der Animation.
Dabei werden **HTML-Elemente** dynamisch verändert.
Animationen können mit oder ohne **JavaScript** erzeugt werden.
Darüber handelt dieses letzte Buchkapitel.

• CSS 2D Transforms

translate(Xpx, Ypx) verschiebt ein Element um X Pixel horizontal und Y Pixel vertikal.
translateX(Npx) verschiebt ein Element um N Pixel horizontal.
translateY(Npx) verschiebt ein Element um N Pixel vertikal.
rotate(±Ndeg) rotiert ein Element um N Winkelgrade in oder gegen den Uhrzeigersinn.
scale(X, Y) ändert die Element-Größe um das X-fache horizontal und das Y-fache vertikal.
scaleX(N) ändert die Element-Größe um das N-fache horizontal.
scaleY(N) ändert die Element-Größe um das N-fache vertikal.
skew(Xdeg, Ydeg) stellt ein Element um einen Winkel schräg - bezogen auf eine Achse.
skewX(Ndeg) stellt ein Element um einen Winkel schräg - bezogen auf die x-Achse.
skewY(Ndeg) stellt ein Element um einen Winkel schräg - bezogen auf die y-Achse.
matrix(p1,p2,p3,p4,p5,p6) kombiniert 6 Transform-Methoden: p1 = scaleX(), p2 = scaleY(), p3 = skewX(), p4 = skewY(), p5 = translateX(), p6 = translateY().

Beispiel:
```
div {
   transform: translate(50px,100px);
   transform: rotate(-90deg);
   transform: scale(2,2);
}
```

• CSS Transitions

Die **transition**-Methode ermöglicht eine sanft-fortschreitende Änderung eines Merkmals.
transition-property: Merkmal ; transition-duration: (Milli)Sekunden;
transition-delay: (Milli)Sekunden; transition-timing-function: Speedcurve;
Für Speedcurve gibt es sechs Parameter (ease, ease-in, ease-out, ease-in-out, linear, bezier) .
Sie bestimmen das Geschwindigkeits-Verhalten der Merkmalsänderung. Beispielsweise wird mit „ease“ der Anfang und das Ende der Merkmalsänderung verzögert.

Beispiel:
```
div {
   transition-property: width;
   transition-duration: 2s;
   transition-delay: 2s;
   transition-timing-function: ease;
}
```

• CSS Animations

Damit werden Animationen ermöglicht auch ohne Verwendung von JavaScript.
Bei allen Animationen wird der CSS-Stil von Elementen graduell verändert.
Dabei muss zuerst ein **Keyframe** der Animation festgelegt werden. Dadurch wird die graduelle Änderung des aktuellen Stils zu einem neuen Stil festgelegt.

Die **animation**-Methode selbst kennt ähnliche Eigenschaften wie die **transition**-Methode (animation-duration, animation-delay und animation-timing-function). Zusätzlich gibt es:
animation-name verweist auf einen keyframe-Bezeichner.
animation-iteration-count bestimmt die Anzahl der Durchführungen („n“ oder „infinite“).
animation-direction bestimmt die Ausrichtung: normal (vorwärts), reverse (rückwärts) , alternate-reverse (vorwärts und rückwärt).
animation-fill-mode bestimmt den Zustand vor bzw. nach der Animation (forwards, backwards).

Beispiel 0:
```
@keyframes mycolor {
   from { background-color: red; }
   to { background-color: blue; }
}

div {
   width:  100px;
   height: 100px;
   background-color: red;
   animation-name: mycolor;
   animation-duration: 2s;
   animation-timing-function: ease;
}
```

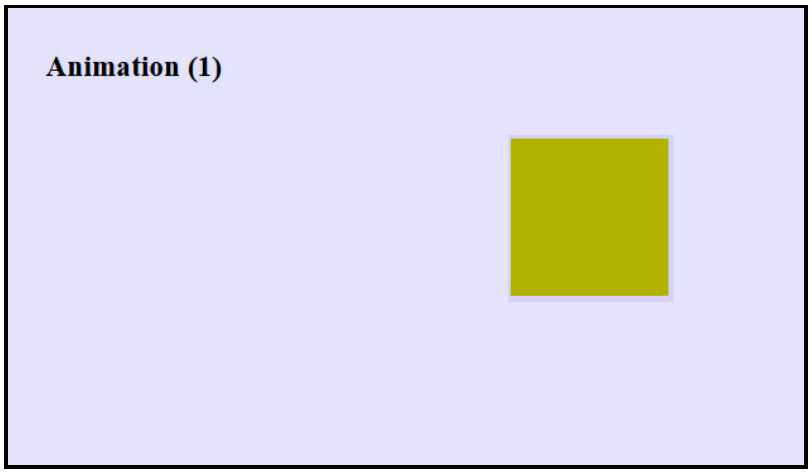
Beispiel 1:
```
@keyframes mymove {
   0%    { background-color: red; left:0px; top:0px; }
   25%   { background-color: green; left:200px; top:0px; }
   50%   { background-color: blue; left:200px; top:200px; }
   75%   { background-color: green; left:0px; top:200px; }
   100% { background-color: red; left:0px; top:0px; }
}

div {
   width:  100px;
   height: 100px;
   position: relative;
   background-color: red;
   animation: mymove 5s infinite;
}
```

Im Beispiel (0) wird ein **div**-Element als quadratische Box (100px) in der Farbe rot definiert.
Eine Animation verändert dann die Farbe durch Verweis auf ein entsprechendes Keyframe.

Im Beispiel (1) wird ein **div**-Element als quadratische Box (100px) in der Farbe rot definiert.
Eine Animation verändert dann die Farbe und die Position durch Verweis auf ein Keyframe.
Dabei wird die Animations-Methode in Kurzform angeschrieben.
Im Keyframe ist anstelle von „from“ – „to“ der Änderungsverlauf in Prozenten angegeben.

Dieses Beispiel (1) wird in ähnlicher Form im Programm „***move01.html***“ realisiert.

[6.1.1.1] „move01.html“



```
<!DOCTYPE html>
<html>
<head>
<title>Animation (1)</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0;" />
<meta name="author" content="H.P">

<style>
body {
  background: #EEEEFF;
  margin-left: 50px;
}

div {
  width: 100px;
  height: 100px;
  background: red;
  position: relative;
  animation: mymove 5s infinite;
}

@keyframes mymove {
  0%   {top: 0px; left: 100px; background: red;}
  25%  {top: 0px; left: 300px; background: blue;}
  50%  {top: 300px; left: 300px; background: yellow;}
  75%  {top: 300px; left: 100px; background: green;}
  100% {top: 0px; left: 100px; background: red;}
}
</style>
</head>

<body>
  <h3>Animation (1)</h3>
  <br>
  <div></div>
  <br>
</body>
</html>
```

• Der Aufruf von Animationen in JavaScript

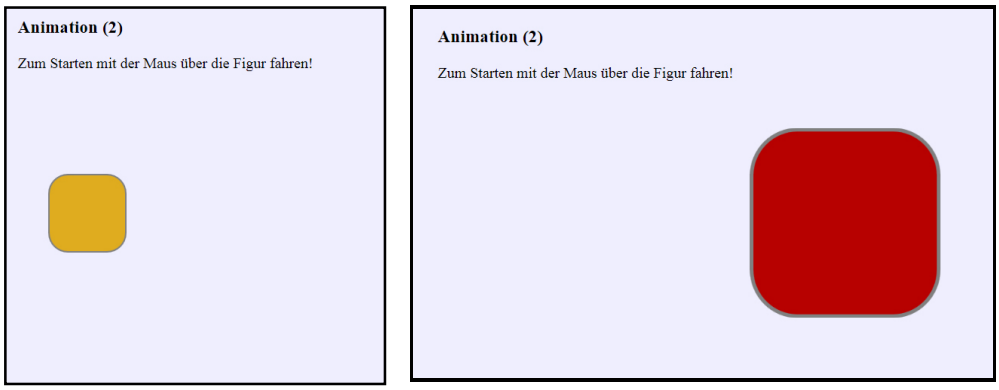
Die **element.animate()**-Methode bindet CSS-Eigenschaften ein, deren Werte dann dynamisch verändert werden. Dabei kann ein Array von Objekten aus CSS-Eigenschaften und deren Werten (d.h. Schlüssel-Wert-Paare) verwendet werden.

Beispiel:
```
element.animate ( [
    { transform : 'translateX(   0px)', color: 'black'; },
    { transform : 'translateX( 100px)', color: 'red'; },
    { transform : 'translateX(   0px)', color: 'black'; },
    ], {
        duration: 3000,
        delay: 2000,
        interations: infinite
      }
);
```

Eine Animation ist ein Ereignis, welches an einem HTML-Element stattfindet. Das Ereignis wird mit einem entsprechenden Event-Handler verarbeitet. Event-Handler-Funktionen können in JavaScript auf unterschiedliche Art codiert werden. Die Zuordnung eines HTML-Elements zu einer JavaScript-Variablen kann mit ***element = document.getElementById('name');*** erfolgen.

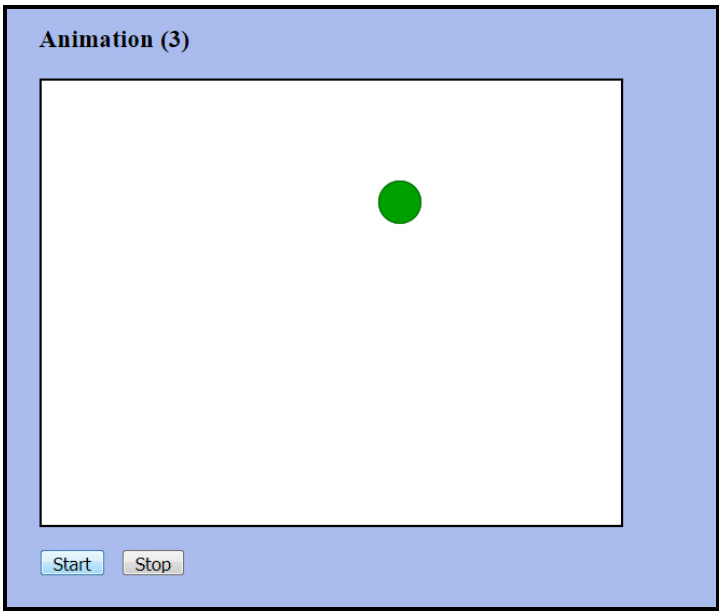
Grundsätzlich kann der gesamte JavaScript-Code in eine **anonyme, selbstausführende Funktion** verpackt werden, welche die entsprechenden Unterfunktionen enthält. Diese Funktion kann nach der Ladung des HTML-Dokuments mit dem „**DOMContentLoaded**"-Ereignis aufgerufen werden.

[6.1.1.2] „move02.html“




```
<!DOCTYPE html>
<html>
<head>
<title>Animation (2)</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0;" />
<meta name="author" content="H.P">

<style>
  body {
    background: #EEEEFF;
    margin-left: 50px;
  }
  #figur {
    position: absolute;
    left: 50px;
    top: 150px;
    width:  50px;
    height: 50px;
    border-radius: 25%;
    background: #DFAC20;
    border: 4px solid gray;
  }
</style>

<script>

  // Aufruf der Funktion "init()" sofort nach Ladung der HTML-Seite
  document.addEventListener("DOMContentLoaded", function() { init(); } );

  function init() {
    var kreis = document.getElementById('figur');
    kreis.onmouseover = function() {
        kreis.animate([
          { transform: 'translate(  0px)' + 'scale(1.0)', background: '#DFAC20' },
          { transform: 'translate(500px)' + 'scale(3.0)', background: 'red' },
          { transform: 'rotate(45deg)' },
          { transform: 'translate(  0px)' + 'scale(1.0)', background: 'black' }
          ],
          { duration: 3000, iterations: 2 }
        );
    };
  }

</script>

</head>

<body>
  <h3>Animation (2)</h3>
  <p>Zum Starten mit der Maus über die Figur fahren!</p>
  <div id="figur"></div>
</body>
</html>
```

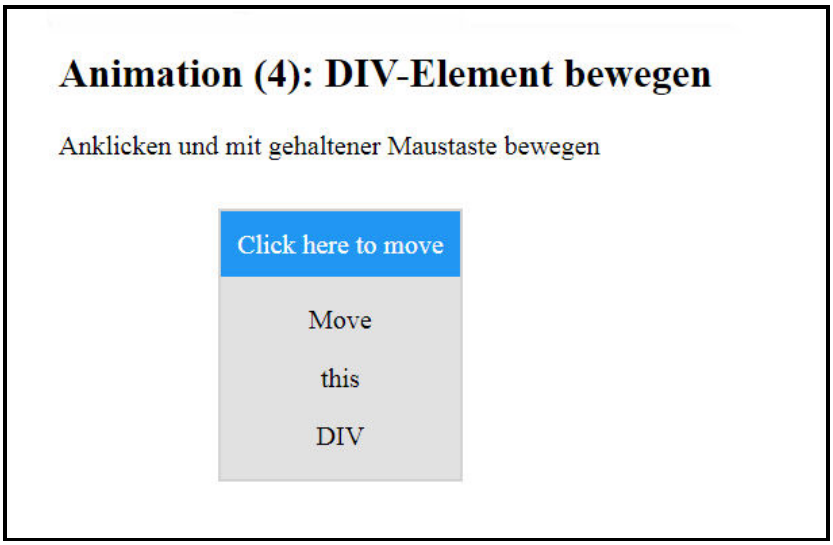
[6.1.1.3] „move03.html“




```
<!DOCTYPE html>
<html>
<head>
<title>Animation (3)</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0;" />
<meta name="author" content="H.P">

<style>
body {
  background: #AABBEE;
  margin-left: 50px;
}
div {
  width:  500px;
  height: 400px;
  background:  white;
  border: 2px solid;
}
#circle {
  background: green;
  left: 50px;
  top:  50px;
  width:  40px;
  height: 40px;
  border-radius: 50%;
  border: 3px solid gray;
}
</style>

<script>

  // Array-Liste von Transformationen
  var move = [
    { transform: 'translate(0, 0)', background: 'green' },
    { transform: 'translate(460px, 360px)', background: 'red' },
    { transform: 'translate(200px, 0px)', background: 'red' },
    { transform: 'translate(0px, 0px)', background: 'green' }
  ];

  function run(){
    var ball = document.getElementById('circle');
    ball.animate(move, {
      duration: 2500,
      iterations: 100,
      fill: 'forwards'
    });
  }
```

```
  function halt() {
    var ball = document.getElementById('circle');
    ball.animate(move, {
      duration: 0,
      iterations: 0,
      fill: 'forwards'
    });
  }

</script>

</head>

<body>
  <h3>Animation (3)</h3>
  <div>
    <div id="circle"></div>
  </div>
  <br>
  <button id="btn1" onclick="run();">Start</button>
    
  <button id="btn2" onclick="halt();">Stop</button>
</body>
</html>
```

[6.1.1.4] „move04.html“




```
<!DOCTYPE html">
<html>
<head>
<title>Animation (4)</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=deventice-width, initial-scale=1.0">
<meta name="author" content="H.P.">

<style>
#mydiv {
  position: absolute;
  background-color: #e1e1e1;
  text-align: center;
  border: 2px solid #d3d3d3;
}
#mydivheader {
  background-color: #2196F3;
  color: #ffffff;
  padding: 10px;
  cursor: move;
}
</style>
```

```
<script>
function dragElement(elem) {
  /* Hauptprogramm mit drei Unterprogrammen */
  var pos1 = 0, pos2 = 0, pos3 = 0, pos4 = 0;
  if (document.getElementById(elem.id + "header")) {
    /* wenn existent, dann ist der Bewegungs-Startpunkt hier */
    document.getElementById(elem.id + "header").onmousedown = dragMouseDown;
  }
  else {
    /* andererseits ist der Bewegungs-Startpunkt irgendwo im DIV-Element */
    elem.onmousedown = dragMouseDown;
  }

  function dragMouseDown(ev) {
    /* Erstes Unterprogramm */
    ev = event || window.event;
    ev.preventDefault();
    pos3 = ev.clientX;   /* Mausposition abspeichern */
    pos4 = ev.clientY;
    document.onmousemove = elementDrag;   /* Funktionsaufruf bei einer Maus-Bewegung */
    document.onmouseup = closeDragElement;
  }

  function elementDrag(ev) {
    /* Zweites Unterprogramm */
    ev = event || window.event;
    ev.preventDefault();
    pos1 = pos3 - ev.clientX;   /* Neue Maus-Position ermitteln */
    pos2 = pos4 - ev.clientY;
    pos3 = ev.clientX;
    pos4 = ev.clientY;
    elem.style.top = (elem.offsetTop - pos2) + "px";   /* Neue Element-Position setzen */
    elem.style.left = (elem.offsetLeft - pos1) + "px";
  }

  function closeDragElement() {
    /* Drittes Unterprogramm */
    document.onmousemove = null;   /* Element-Bewegung beenden */
    document.onmouseup = null;
  }
}
</script>
</head>

<body>
<h2>Animation (4): DIV-Element bewegen</h2>
<p>Anklicken und mit gehaltener Maustaste bewegen</p>

<div id="mydiv">
  <div id="mydivheader">Click here to move</div>
  <p>Move</p>
  <p>this</p>
  <p>DIV</p>
</div>

<script>
   dragElement(document.getElementById("mydiv"));
</script>

</body>
</html>
```

Hinweis 1: Die Merkmale ***offsetLeft*** und ***offsetTop*** geben die originalen Positionswerte eines Elementes an (relativ zum jeweiligen Eltern-Element).

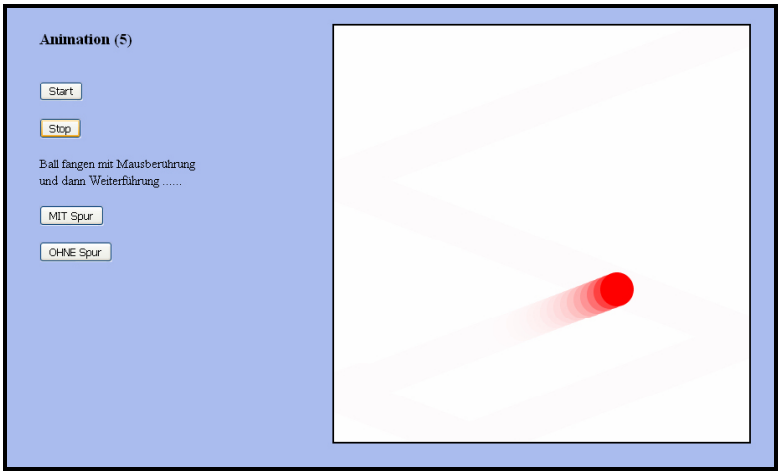
Hinweis 2: Die Variable ***null*** ist ein leeres Objekt. Ihre Zuweisung zu einer Funktion bewirkt nichts, d.h. die Funktion wird nicht ausgeführt.

[6.1.1.5] „move05.html“

Hinweis: Ein Objekt ***ball*** bewegt sich auf einem Zick-Zack-Weg in einem <***canvas***>-Bereich.
Mit den Schaltern <btn1> und <btn2> kann die Ballbewegung gestartet und gestoppt werden.
Mit den Schaltern <btn3> und <btn4> kann eine Bewegungsspur ein- und ausgeblendet werden.
Im Canvas-Bereich wird das Ereignis ***mouseover*** abgefragt. Wenn dieses Ereignis genau auf dem Ort des sich bewegenden Balls erfolgt, dann wird der Ball mit gehaltener Maustaste bewegt.

Im Canvas-Bereich wird die Farbe mit ***rgb(255,255,255)*** auf weiß gesetzt - ohne Transparenz. Verwendet man den Befehl ***rgba(255,255,255,0.25)***, dann steuert der vierte Parameter die Farbtransparenz (0 = totale Tranzparenz, 1 = totale Deckkraft). Durch den Wert 0.25 ist die Deckkraft des Canvas vermindert, sodass ein Nachzieh-Effekt (eine Spur) der Ballbewegung erreicht wird.

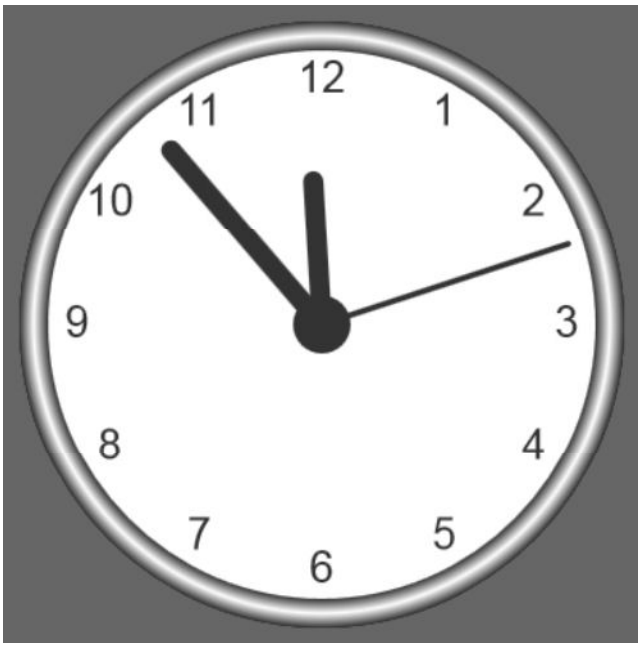
Zu erwähnen ist noch die Methode ***raf = requestAnimationFrame(draw)***, welche im Argument die Animation ***draw*** aufweist. Sie bewirkt ähnlich wie ***setTimeout(draw, time)*** eine dauernde Wiederholung von ***draw***, wenn sie innerhalb von ***draw*** aufgerufen wird. Mit ***cancelAnimationFrame(raf)*** werden diese Aufrufe beendet.




```
<!DOCTYPE html>
<html>
<head>
<title>Animation (5)</title>
<meta charset="ISO-8859-1">
<meta name="author" content="H.P">

<style>
  body {
    background: #AABBEE;
    margin-left: 50px;
  }
</style>

<script>
var canvas;
var ctx;
var raf;
var running = false;
var spur = false;

function initCanvas() {
  canvas = document.getElementById('canvas');
  ctx = canvas.getContext('2d');
  clear();
}

function clear() {
  if ( spur ) { ctx.fillStyle = 'rgba(255, 255, 255, 0.25)'; }
  else { ctx.fillStyle = 'rgb(255, 255, 255)'; }
  ctx.fillRect(0,0,canvas.width,canvas.height);
  ctx.lineWidth = 4;
  ctx.strokeStyle = "black";
  ctx.strokeRect(0,0,500,500);
}
```

```
var ball = {
  x: 100,
  y: 100,
  vx: 8,
  vy: 3,
  radius: 20,
  color: 'red',
  paint: function() {
    ctx.beginPath();
    ctx.arc(this.x, this.y, this.radius, 0, Math.PI * 2, true);
    ctx.fillStyle = this.color;
    ctx.fill();
    ctx.closePath();
  }
}

function draw() {
  clear();
  ball.paint();
  ball.x += ball.vx;
  ball.y += ball.vy;
  if (ball.y + ball.vy > canvas.height || ball.y + ball.vy < 0) {
    ball.vy = -ball.vy;
  }
  if (ball.x + ball.vx > canvas.width || ball.x + ball.vx < 0) {
    ball.vx = -ball.vx;
  }
  raf = window.requestAnimationFrame(draw);
}

function move(e) {
  clear();
  r = 20;
  x = e.clientX - 400;
  y = e.clientY - 40;
  if ( (Math.abs(ball.x - x) < r) && (Math.abs(ball.y - y) < r) ) {
    stop();
    ball.x = x;
    ball.y = y;
    ball.paint();
  }
}

function run() {
  if (!running) {
    raf = window.requestAnimationFrame(draw);
    running = true;
  }
}

function stop() {
  window.cancelAnimationFrame(raf);
  running = false;
}

function spurON()  { spur = true; }
function spurOFF() { spur = false; }

</script>
</head>

<body onload="initCanvas()">
<h3>Animation (5)</h3>
<div style="position:absolute; top:40px; left:400px">
  <canvas id='canvas' width='500' height='500' onmousemove = "move(event)">
     Your browser does not support HTML5 Canvas.
  </canvas>
</div>
 <br>
<button id="btn1" onclick="run()"> Start </button>
<br><br>
<button id="btn2" onclick="stop()"> Stop </button>
<br><br>
Ball fangen mit Mausberührung<br>
und dann Weiterführung ......
<br><br>
<button id="btn3" onclick="spurON()"> MIT Spur </button>
<br><br>
<button id="btn4" onclick="spurOFF()"> OHNE Spur </button>
</body>
</html>
```

[6.1.2] Eine einfache Wanduhr (clock.html)



Das Programm "**clock.html**" simuliert eine einfache Wanduhr auf einem <canvas>-Objekt.
Dabei wird die vordefinierte Grafikfunktion **createRadialGradient(p1, p2, p3, p4, p5, p6)** zur Darstellung des Uhrenrahmens verwendet. Damit wird ein radial/zirkulärer Farbverlauf (Gradient) von grafischen Objekten (Rechtecke, Kreise, Linien, Texte, etc.) erzeugt, und zwar als Werte von **strokeStyle** und **fillStyle**. Die sechs Parameter haben folgende Bedeutung:

p1 … x-Koordinate des Startkreises des Gradienten
p2 … y-Koordinate des Startkreises des Gradienten
p3 … Radius des Startkreises
p4 … x-Koordinate des Endkreises des Gradienten
p5 … y-Koordinate des Endkreises des Gradienten
p6 … Radius des Endkreises

Mit dem zusätzlichen Befehl **addColorStop(p1, p2)** kann eine bestimmte Farbe (**p2**) und ihr jeweiliger Radius (**p1**) im Gradienten spezifiziert werden.

Der Befehl **textBaseline = "position"** bestimmt in **fillText** die Position des Textes relativ zur jeweiligen horizontalen Linie (bottom, middle oder top).

```
<!DOCTYPE html>
<html>
<head>
<title>clock.html</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Uhrendarstellung">
<meta name="author" content="Herbert Paukert">
</head>

<body style="background-color:antiquewhite">
<canvas id="canvas" width="400" height="400" style="background-color:#666"> </canvas>

<script>
  var canvas = document.getElementById("canvas");
  var ctx = canvas.getContext("2d");
  var radius = canvas.height / 2;
  ctx.translate(radius, radius);
  radius = radius * 0.90
  setInterval(drawClock, 1000);
```

```
  function drawClock() {
  // Hauptfunktion
    drawFace(ctx, radius);
    drawNumbers(ctx, radius);
    drawTime(ctx, radius);
  }

  function drawFace(ctx, radius) {
  // Uhrfläche
    var grad;
    ctx.beginPath();
    ctx.arc(0, 0, radius, 0, 2*Math.PI);
    ctx.fillStyle = 'white';
    ctx.fill();
    grad = ctx.createRadialGradient(0,0,radius*0.95, 0,0,radius*1.05);
    grad.addColorStop(0, '#333');
    grad.addColorStop(0.5, 'white');
    grad.addColorStop(1, '#333');
    ctx.strokeStyle = grad;
    ctx.lineWidth = radius*0.1;
    ctx.stroke();
    ctx.beginPath();
    ctx.arc(0, 0, radius*0.1, 0, 2*Math.PI);
    ctx.fillStyle = '#333';
    ctx.fill();
  }

  function drawNumbers(ctx, radius) {
  // Uhrzeiten beschriften
    var ang;
    var num;
    ctx.font = radius*0.15 + "px Arial";
    ctx.textBaseline="middle";
    ctx.textAlign="center";
    for(num = 1; num < 13; num++){
      ang = num * Math.PI / 6;
      ctx.rotate(ang);
      ctx.translate(0, -radius*0.85);
      ctx.rotate(-ang);
      ctx.fillText(num.toString(), 0, 0);
      ctx.rotate(ang);
      ctx.translate(0, radius*0.85);
      ctx.rotate(-ang);
    }
  }

  function drawTime(ctx, radius){
  // Uhrzeiger einstellen
    var now = new Date();
    var hour = now.getHours();
    var minute = now.getMinutes();
    var second = now.getSeconds();
    //hour
    hour=hour%12;
    hour=(hour*Math.PI/6)+
    (minute*Math.PI/(6*60))+
    (second*Math.PI/(360*60));
    drawHand(ctx, hour, radius*0.5, radius*0.07);
    //minute
    minute=(minute*Math.PI/30)+(second*Math.PI/(30*60));
    drawHand(ctx, minute, radius*0.8, radius*0.07);
    // second
    second=(second*Math.PI/30);
    drawHand(ctx, second, radius*0.9, radius*0.02);
  }

  function drawHand(ctx, pos, length, width) {
  // Uhrzeiger
    ctx.beginPath();
    ctx.lineWidth = width;
    ctx.lineCap = "round";
    ctx.moveTo(0,0);
    ctx.rotate(pos);
    ctx.lineTo(0, -length);
    ctx.stroke();
    ctx.rotate(-pos);
  }
</script>

</body>
</html>
```

[6.1.3] Zwei Reaktionsspiele

• Das Reaktionsspiel „escape.html“

Das Programm „**escape.html**“ ist ein Reaktionsspiel, dessen Spielfeld ein -Objekt ist, ausgestattet mit der Eigenschaft *opacity: 0.5* in der CSS-Anweisung *#field*. Durch den Wert 0.5 ist die Deckkraft (Transparenz) zu 50% durchsichtig, sodass die anderen Spielobjekte sichtbar sind.

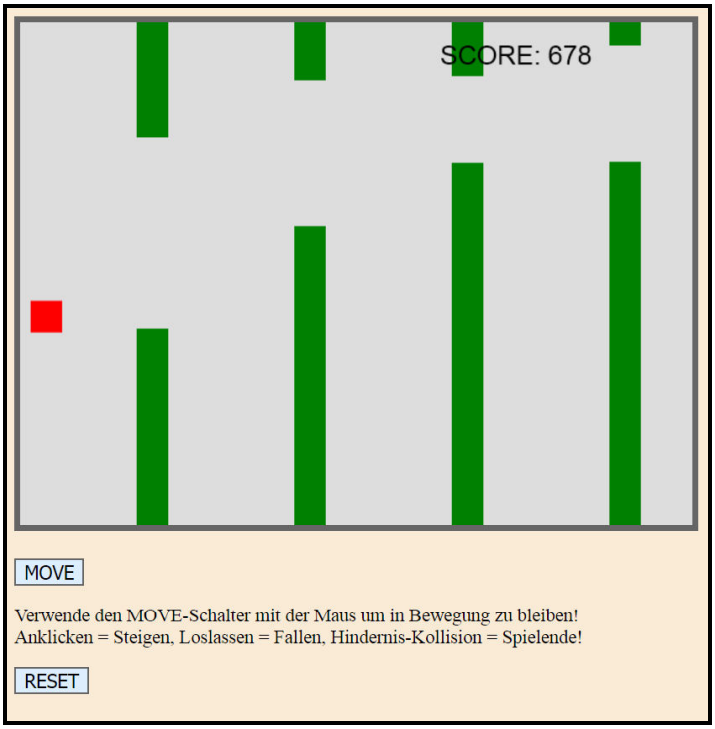

```
<html>
<head>
<title>Reaktionstest ESCAPE, Version 5.0</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=0.75">
<meta name="author" content="Herbert Paukert">

<style>
  #base   { background-color:#DDDDDD; color:#000000; margin: 0px 0px 0px 0px; }
  #field  { position: absolute; left: 0px; top:0px; width:350px; height:350px;
            background:white; border: 50px solid black; opacity:0.25; }
  #box    { position: absolute; left: 205px; top: 205px; width: 40px; height: 40px;
            background-color: #FF0000; }
  #enemy0 { position: absolute; left: 270px; top: 60px; width: 60px; height: 50px;
            background-color: #0000FF; }
  #enemy1 { position: absolute; left: 290px; top: 330px; width: 100px; height: 20px;
            background-color: #0000FF; }
  #enemy2 { position: absolute; left: 70px; top: 320px; width: 30px; height: 60px;
            background-color: #0000FF; }
  #enemy3 { position: absolute; left: 70px; top: 70px; width: 60px; height: 60px;
            background-color: #0000FF; }
  #info   { position:absolute; left:500px; top:10px; width:350px;
            font-family:sans-serif; font-size:16px; border:1px solid blue;
            color:black; background:antiquewhite; }
  #velo   { position: absolute; left: 500px; top:390px;
            background:white; border: 1px solid black; }
</style>

<script>
var curX, curY, curX2, curY2, boxX, boxY;     // Aktuelle Maus- und Box-Positionen
var moving = 0, touch = 0;                    // Steuervariable für Bewegung und Kollision
var gametime = 0, started = 0, speed = 100;   // Steuervariable für Zeit und Geschwindigkeit
var starttime, endtime, finaltime = 0;
var enemyxdir = new Array(1,1,1,1);           // Bewegungsrichtung
var enemyydir = new Array(1,1,1,1);
var sstep = 0;                                // Geschwindigkeits-Stufe

document.onmousedown = start;
document.onmousemove = checkLocation;

function startclock() {var today = new Date(); starttime = today.getTime();}
function endclock()   {var today = new Date(); endtime = today.getTime();}
function calctime()   {var time = (endtime - starttime)/1000; return time;}

function setposX(divname, xpos) { document.getElementById(divname).style.left = xpos; }
function setposY(divname, ypos) { document.getElementById(divname).style.top = ypos; }
```

```
function getposX(divname) {
  var divData = document.getElementById(divname).getBoundingClientRect();
  return divData.left;
}

function getposY(divname) {
  var divData = document.getElementById(divname).getBoundingClientRect();
  return divData.top;
}

function getsize(divname, dimension) {
  var divData = document.getElementById(divname).getBoundingClientRect();
  var divsize = 0;
  if (dimension == 'y') { divsize = divData.height; }
  else if (dimension == 'x') { divsize = divData.width; }
  return divsize;
}

function checktouching(num) {
// check to see if 'box' is touching 'enemies'
// set touch = 1 if it is touching an enemy
  var enemy = "enemy" + num;
  var difX = getposX('box') - getposX(enemy);
  var difY = getposY('box') - getposY(enemy);
  if (difX > (-1 * getsize('box', 'x')) && difX < getsize(enemy, 'x') &&
      difY > (-1 * getsize('box', 'y')) && difY < getsize(enemy, 'y')) {touch = 1;}
  else {touch = 0;}
}

function movenemy(num, step_x, step_y){
// move the enemy "num"
  var enemy = "enemy" + num;
  var enemyx = getsize(enemy, 'x');
  var enemyy = getsize(enemy, 'y');
  if (getposX(enemy) >= (450 - enemyx) || getposX(enemy) <= 0) {
     enemyxdir[num] = -1 * enemyxdir[num];
  }
  if (getposY(enemy) >= (450 - enemyy) || getposY(enemy) <= 0) {
     enemyydir[num] = -1 * enemyydir[num];
  }
  var newposx = getposX(enemy) + (step_x * enemyxdir[num]);
  var newposy = getposY(enemy) + (step_y * enemyydir[num]);
  setposX(enemy, newposx);
  setposY(enemy, newposy);
  checktouching(num);
  if (touch == 1) { reset(); }
}

function movenemies() {
// increase the time and move all enemies
  gametime = gametime + 1;
  movenemy(0,-10,12);
  movenemy(1,-12,-20);
  movenemy(2,15,-13);
  movenemy(3,17,11);
  setTimeout(movenemies,speed);
}

function start(event) {
// start the game
  var btn = document.getElementsByName("rb");
  for (var i = 0; i < btn.length; i++) {
      if (btn[i].checked) { sstep = btn[i].value; }
  }
  speed = 100 - 15 * parseInt(sstep);
  var fdata = document.getElementById('field').getBoundingClientRect();
  var fw = fdata.width - 50;
  curX = event.pageX;
  curY = event.pageY;
  if (curX > fw || curY > fw) { return false;}
  if (started == 0) { movenemies(); startclock(); started = 1; }
  curX2 = curX - 40;
  curY2 = curY - 40;
  boxX = curX - 20;
  boxY = curY - 20;
  var boxleft = getposX("box");
  var boxtop  = getposY("box");
  if (curX > boxleft && curX2 < boxleft && curY > boxtop && curY2 < boxtop) {
     moving = 1;
     setposX('box', boxX);
     setposY('box', boxY);
  }
}
```

```
function reset(){
// show the result and start a new game
  endclock();
  moving = 0;
  if (finaltime == 0) {
     finaltime = calctime();
     alert('Spielzeit: ' + finaltime + ' Sekunden' + '\nGeschwindigkeitsstufe: ' + sstep);
     document.location.reload();
  }
}

function checkLocation(event) {
// check the positions
  curX = event.pageX;
  curY = event.pageY;
  boxX = curX - 20;
  boxY = curY - 20;
  checktouching('1');
  if (moving == 1 && touch == 0){
     setposX('box',boxX);
     setposY('box',boxY);
     if (curY > 70 && curX > 70 && curY < 380 && curX < 380) { return false; }
     else { reset(); }
  }
  else if (touch == 1) { reset(); }
}

</script>
</head>

<body id="base">
<form>

<div id="box"> </div>
<div id="enemy0"> </div>
<div id="enemy1"> </div>
<div id="enemy2"> </div>
<div id="enemy3"> </div>

<!-- Testfeld, ca. mit einer Breite und Höhe von 9 Quadrat-Boxen (9 x 40 = 360) -->
<div id="field"> </div>

<div id="velo">
<br>  Geschwindigkeit 
<input type="radio" name="rb" value="0"> 0 
<input type="radio" name="rb" value="1"> 1 
<input type="radio" name="rb" value="2"> 2 
<input type="radio" name="rb" value="3"> 3 
<input type="radio" name="rb" value="4"> 4 
<input type="radio" name="rb" value="5"> 5 
   <br> 
</div>

<fieldset id="info">
  <br>
<b>  Reaktionstest «ESCAPE»<br><br></b>
  Klicke mit der Maus auf das rote Quadrat<br>
  und halte es.<br><br>
  Bewege das Quadrat ohne mit den blauen<br>
  Figuren zusammenzustoßen. Berühre auch<br>
  nicht den Feldrahmen.<br><br>
  Wenn Du mehr als 20 Sekunden schaffst,<br>
  dann ist das SEHR GUT.<br><br>
  Zusätzlich können sechs Geschwindigkeits-<br>
  stufen eingestellt werden !<br><br>
  &copy; Herbert Paukert, 2019, Version 5.0<br>
  <br>
</fieldset>

</form>
</body>
</html>
```

• Das Reaktionsspiel „reago.html“

Das Programm „**reago.html**“ ist ein Reaktionsspiel, dessen Spielfeld ein <canvas>-Objekt ist, auf dem verschiedene Hindernisse erzeugt werden, welche sich schrittweise nach links verschieben. Mit dem Schalter [MOVE] können Auf- und Abwärtsbewegungen einer quadratischen Spielfigur gesteuert werden. Ein Anklicken des Schalters mit der Maus bewegt die Figur hinauf, ein Loslassen der Maus bewirkt eine Abwärtsbewegung, wobei sich die Bewegungsgeschwindigkeit zusätzlich automatisch ändert. Eine Kollision mit den Hindernissen ist unbedingt zu vermeiden. Das Spiel ist dann beendet, wenn die Spielfigur mit einem Hindernis kollidiert. Mit dem Schalter [RESET] wird ein neues Spiel gestartet.



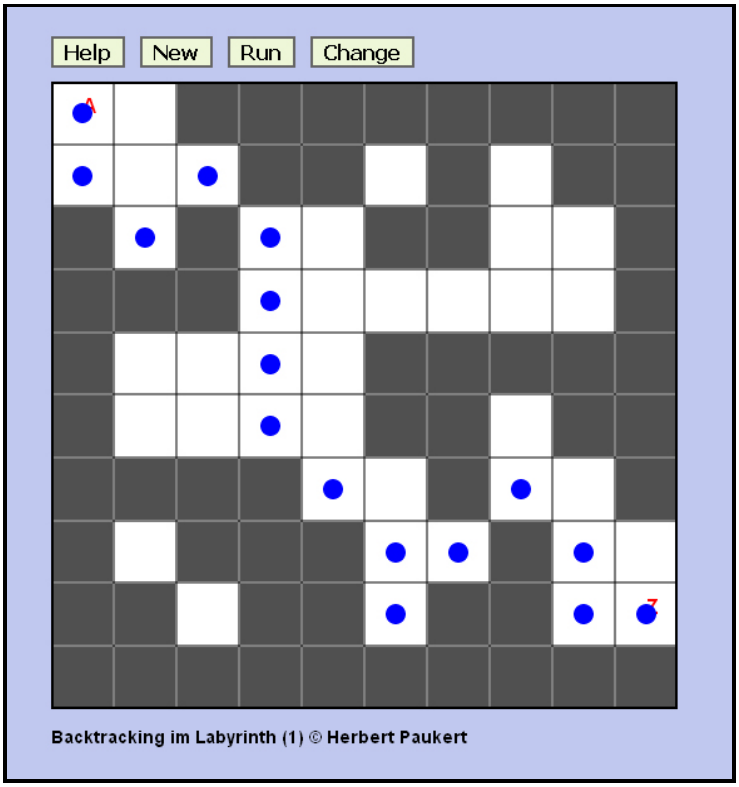

```
<!DOCTYPE html>
<html>
<head>
<title>reago.html</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Reaktionsspiel">
<meta name="author" content="Herbert Paukert">
<style>
   body   { color: black; background-color: antiquewhite; font-size: 18px; }
   canvas { border:6px solid #666; background-color: #ddd; }
   button { border:2px solid #666; background-color: #def; font-size: 18px; }
</style>
</head>

<body onload="startGame()">

<script>
  var myGamePiece;
  var myObstacles = [];
  var myScore;

  function startGame() {
  // Hauptfunktion
    myGamePiece = new component(30, 30, "red", 10, 200);
    myGamePiece.move = 0.05;
    myScore = new component("25px", "Arial", "black", 400, 40, "text");
    myGameArea.start();
  }
```

```
var myGameArea = {
// Spielfeld
  canvas: document.createElement("canvas"),
  start : function() {
      this.canvas.width  = 640;
      this.canvas.height = 480;
      this.context = this.canvas.getContext("2d");
      document.body.insertBefore(this.canvas, document.body.childNodes[0]);
      this.frameNo = 0;
      this.interval = setInterval(updateGameArea, 20);
  },
  clear : function() {
      this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
  }
}

function component(width, height, color, x, y, type) {
// Spielobjekt
  this.type = type;
  this.score = 0;
  this.width = width;
  this.height = height;
  this.speedX = 0;
  this.speedY = 0;
  this.x = x;
  this.y = y;
  this.move = 0;
  this.moveSpeed = 0;
  this.update = function() {
      ctx = myGameArea.context;
      if (this.type == "text") {
          ctx.font = this.width + " " + this.height;
          ctx.fillStyle = color;
          ctx.fillText(this.text, this.x, this.y);
      } else {
          ctx.fillStyle = color;
          ctx.fillRect(this.x, this.y, this.width, this.height);
      }
  }
  this.newPos = function() {
      if (this.y < 5) { this.y = 200; this.moveSpeed = this.move; }
      this.moveSpeed += this.move;
      this.x += this.speedX;
      this.y += (this.speedY + this.moveSpeed);
      this.hitBottom();
  }
  this.hitBottom = function() {
      var rockbottom = myGameArea.canvas.height - this.height;
      if (this.y > rockbottom) {
          this.y = rockbottom;
          this.moveSpeed = 0;
      }
  }
  this.crashWith = function(otherobj) {
      var myleft = this.x;
      var myright = this.x + (this.width);
      var mytop = this.y;
      var mybottom = this.y + (this.height);
      var otherleft = otherobj.x;
      var otherright = otherobj.x + (otherobj.width);
      var othertop = otherobj.y;
      var otherbottom = otherobj.y + (otherobj.height);
      var crash = true;
      if ((mybottom < othertop) || (mytop > otherbottom) || (myright < otherleft) ||
          (myleft > otherright)) { crash = false; }
      return crash;
  }
}

function updateGameArea() {
// Spielfeld-Erneuerung
  var x, height, gap, minHeight, maxHeight, minGap, maxGap;
  for (i = 0; i < myObstacles.length; i += 1) {
      if (myGamePiece.crashWith(myObstacles[i])) {
          return;
      }
  }
```

```
    myGameArea.clear();
    myGameArea.frameNo += 1;
    if (myGameArea.frameNo == 1 || allintervals(150)) {
        x = myGameArea.canvas.width;
        minHeight = 20;
        maxHeight = 200;
        height = Math.floor(Math.random()*(maxHeight-minHeight+1)+minHeight);
        minGap = 50;
        maxGap = 200;
        gap = Math.floor(Math.random()*(maxGap-minGap+1)+minGap);
        myObstacles.push(new component(30, height, "green", x, 0));
        myObstacles.push(new component(30, (x - height - gap), "green", x, (height + gap)));
    }
    for (i = 0; i < myObstacles.length; i += 1) {
        myObstacles[i].x += -1;
        myObstacles[i].update();
    }
    myScore.text="SCORE: " + myGameArea.frameNo;
    myScore.update();
    myGamePiece.newPos();
    myGamePiece.update();
  }

  function allintervals(n) {
    if ( (myGameArea.frameNo / n) % 1 == 0 ) {return true;}
    return false;
  }

  function movePiece(n) { myGamePiece.move = n; }

  function reset() { location.reload(); }

</script>

<br><br>
<button onmousedown="movePiece(-0.10)" onmouseup="movePiece(0.05)"> MOVE </button>
<p>Verwende den MOVE-Schalter mit der Maus um in Bewegung zu bleiben!<br>
Anklicken = Steigen, Loslassen = Fallen, Hindernis-Kollision = Spielende!
</p>
<button onclick="reset()"> RESET </button>
<br>

</body>
</html>
```

[6.1.4] Wegsuche im Labyrinth

- **Automatisches Backtracking im Labyrinth (laby1.html)**



Mit [*New*] wird ein Labyrinth mit zufälligen Hindernissen (schwarz) erzeugt. Die freien Wegfelder sind weiß. Das mit "A" markierte Feld ist der Eingang, das mit "Z" markierte Feld ist der Ausgang im Labyrinth. Mit [*Run*] startet das Programm die Suche nach einem freien Weg von Eingang bis Ausgang. Die Suche erfolgt mit rekursivem Backtracking.

Hinweis 1: Wenn die Hindernisse so dicht liegen, dass kein freier Durchgang mehr möglich ist, dann kann das Labyrinth nachträglich mit [*Change*] geändert werden. Dabei werden innerhalb des Labyrinths mit Mausklick Hindernisse entfernt oder hinzugefügt.

Hinweis 2: Im rekursiven Backtracking startet der Computer zuerst beim Startfeld (A,0) links oben. Er versucht nun die Nachbarfelder zu betreten, beginnend mit dem direkt darunter liegenden Feld. Er wandert dann gegen den Uhrzeigersinn in alle acht Richtungen, wenn das jeweilige Nachbarfeld ein Hindernis (schwarz,1) ist. Ist das Nachbarfeld frei (weiß,0), so wird das Ausgangsfeld als Spur (track,3) markiert und dann das freie Nachbarfeld betreten. Dieses Verfahren wird immer wiederholt. Trifft der Computer dabei auf ein Hindernis oder auf seine eigene Spur, dann springt er zu jenem Feld zurück (back), wo die letzte Richtungsänderung stattgefunden hat. Dort wird das Verfahren dann fortgesetzt. Dadurch wird ein Ariadne-Faden im Labyrinth simuliert. Durch Änderung der Reihenfolge der Richtungswechsel können sich beim Backtracking auch der Verlauf und die Länge des gefundenen Weges ändern.

Das Verfahren ist erfolgreich beendet, wenn das Zielfeld (Z,2) rechts unten erreicht wird. Das Verfahren ist dann nicht erfolgreich beendet, wenn kein freies Feld mehr betreten werden kann.

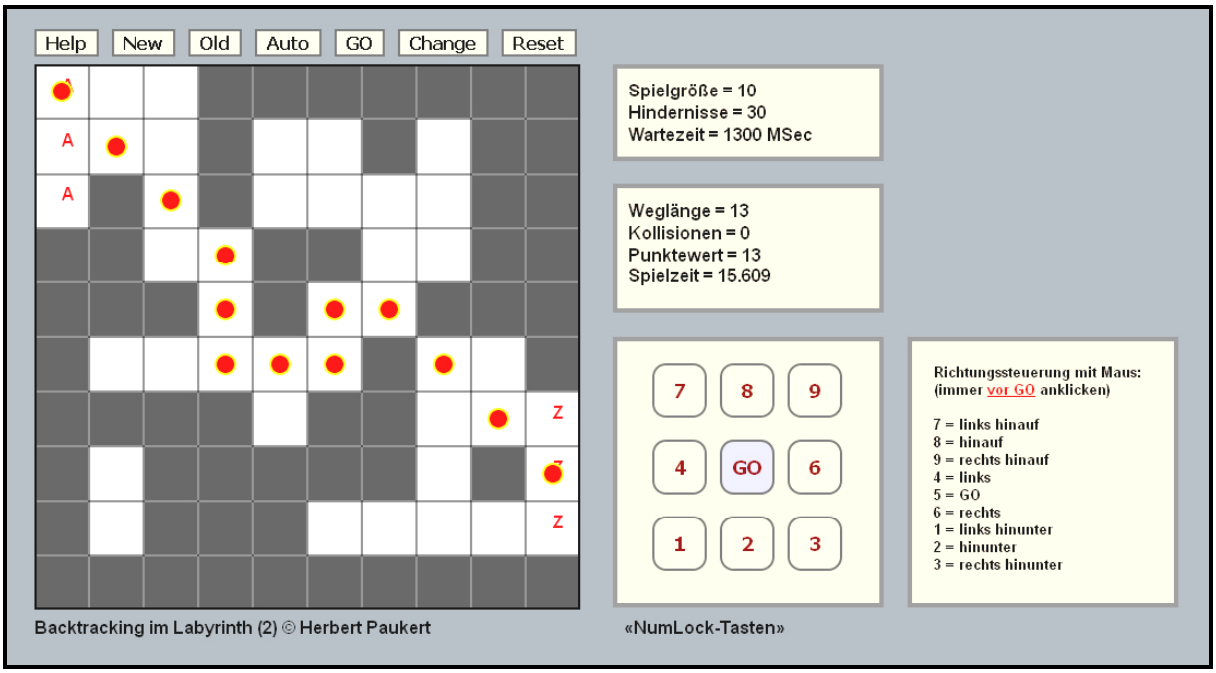
```
<!doctype html>
<html>
<head>
  <title>Backtracking im Labyrinth (laby1)</title>
  <meta charset="ISO-8859-1">
  <meta name="description" content="Backtracking im Labyrinth">
  <meta name="author" content="Herbert Paukert">

  <style>
    body { background-color:#C0C8EF; color:black;
           font-family:Arial; font-size:14px; font-weight:bold; text-size-adjust: none;
           text-align:left; margin-left:60px; }
    .cd  { border:2px solid #666666; background-color:#EEF8D8; font-size:16px; font-weight:bold;}
    .cd1 { border:2px solid #666666; background-color:#EEF8D8; color: darkred;
           font-size:18px; font-weight:bold; width: 50px; height: 50px; border-radius:25%;}
    #div1 { position:absolute; left:660px; top:60px; visibility:hidden; }
    #hp   { position:absolute; left:60px; top:560px; }
  </style>

  <script>

    /* globale Variable (Anfang) ------------------------------------------ */

    var canvas;              //  grafische Leinwand (Spielfeld)
    var ctx;                 //  Kennzeichen der Leinwand

    var len = 500;           //  Abmessungen des quadratischen Rasters
    var min = 10;            //  Minimale Rasterzeilen (bzw. Spalten)
    var max = 16;            //  Maximale Rasterzeilen (bzw. Spalten)
    var d = len / max;       //  Größe eines Rasterfeldes
    var e = d / 2;           //  Halbe Länge eines Rasterfeldes
    var anz = 30;            //  Anzahl der Hindernisse (30  bis 90)

    var neu = false;         //  Steuervariable für ein neues Labyrinth
    var change = false;      //  Steuervariable für Farbe (weiß - schwarz)
    var help = false;        //  Steuervariable für Hilfe

    var okay;                //  Steuervariable für das Backtracking
    var count;               //  Schrittzähler beim Backtracking

    var mat = new Array();   //  Matrix mat[i][j], i = Zeile, j = Spalte
    for (i = 0; i <= max; i++) { mat[i] = new Array(max); }
                             //  mögliche Feldwerte (0,1,2 und 3):
                             //  0 = freies Wegfeld (weiß)
                             //  1 = Hindernis- oder Randfeld (schwarz)
                             //  2 = freies Zielfeld (weiß)
                             //  3 = Feldmarkierung beim Backtracking

    var mp = new Array();    // Mittelpunkte der Rasterfelder
    for (i = 0; i <= max; i++) { mp[i] = new Array(max); }

    function TPoint(tx,ty)   // Punkt im Labyrinth
    {
      this.x = tx;
      this.y = ty;
    }
    var P = new TPoint;

    /* globale Variable (Ende) -------------------------------------------- */


    function myTrim(x) { return x.replace(/^\s+|\s+$/gm,'') }

    function drawM() {
    /* Mittelpunkte der Rasterfelder bestimmen */
      for (i = 1; i <= max; i++) {
        for (j = 1; j <= max; j++) {
            a = (i-1)*d + e; b = (j-1)*d + e; mp[i][j] = new TPoint(b,a);
            mat[i][j] = 0;
        }
      }
    }

    function initMatrix(anz) {
    /* Matrix initialisieren */
      for (i = 1; i <= max; i++) {
        for (j = 1; j <= max; j++) { mat[i][j] = 0; }
      }
```

```
  for (i = 1; i <= max; i++) {
      mat[i][1] = 1;
      mat[i][max] = 1;
  }
  for (j = 1; j <= max; j++) {
      mat[1][j] = 1;
      mat[max][j] = 1;
  }
  p = 0;
  n = max * max - 40;
  while (p < anz ) {
    if (p > n ) { anz = n; break; }
    i = Math.floor(max * Math.random() ) + 1;
    j = Math.floor(max * Math.random() ) + 1;
    if (mat[i][j] != 1)  {
       p = p + 1;
       mat[i][j] = 1;
    }
  }
  mat[1][1] = 0; mat[1][2] = 0;
  mat[2][1] = 0; mat[2][2] = 0;
  mat[max-2][max] = 0;
  mat[max-1][max] = 2;
}

function showMatrix(max) {
/* Matrix anzeigen */
  change = false;
  for (i = 1; i <= max; i++) {
    for (j = 1; j <= max; j++) {
         z = mat[i][j];
         if (z == 1) { drawRect(mp[i][j]); }
         drawText(' ',mp[i][j]);
    }
  }
  drawText('A',mp[1][1]);
  drawText('Z',mp[max-1][max]);
  drawBorder();
}

function newGame() {
/* Neues Labyrinth initialisieren */
  initCanvas();
  neu = true;
  zahl = min;
  zahl = prompt('Größe des Labyrinth (10 - 16)',zahl);
  zahl = myTrim(zahl);
  if (!isNaN(zahl)) {
    if ((zahl < 10) || ( zahl > 16)) { zahl = 12; }
  }
  else { zahl = 12; }
  max = zahl;
  min = zahl;
  zahl = anz;
  zahl = prompt('Maximale Hindernis-Anzahl (20 - 90)',zahl);
  zahl = myTrim(zahl);
  if (!isNaN(zahl)) {
    if ((zahl < 20) || ( zahl > 90)) { zahl = 30; }
  }
  else { zahl = 30; }
  anz = zahl;
  d = len / max;
  e = d / 2;
  drawField();
  drawM();
  initMatrix(anz);
  showMatrix(max);
}

function initCanvas() {
/* Canvas initialisieren */
   canvas = document.getElementById('MyCanvas');
   ctx = canvas.getContext('2d');
   ctx.lineWidth = 4;
   ctx.fillStyle = "white";
   ctx.fillRect(0,0,len,len);
   ctx.strokeStyle = "black";
   ctx.strokeRect(0,0,len,len);
   ctx.lineWidth = 2;
}
```

```
function drawField() {
/* Raster zeichnen */
   ctx.lineWidth = 1;
   ctx.strokeStyle = "black";
   ctx.beginPath();
   for (i = 0; i < max; i++) { ctx.moveTo(0,i*d); ctx.lineTo(len,i*d); }
   for (i = 0; i < max; i++) { ctx.moveTo(i*d,0); ctx.lineTo(i*d,len); }
   ctx.stroke();
}

function drawBorder() {
/* Rahmen zeichnen */
   ctx.lineWidth = 4;
   ctx.strokeStyle = "black";
   ctx.strokeRect(0,0,500,500);
}

function drawRect(M) {
/* Ein Rasterquadrat mit Mittelpunkt M zeichnen und einfärben */
   r = e-1
   var l = (M.x - r);
   var t = (M.y - r);
   var w = 2*r;
   var h = 2*r;
   if (!change) { ctx.fillStyle = "rgb(80,80,80)"; }
   else { ctx.fillStyle = "rgb(255,255,255)"; }
   ctx.fillRect(l,t,w,h);
}

function drawBall(M) {
/* Einen Kreis mit Mittelpunkt M zeichnen und einfärben */
   r = 8;
   ctx.beginPath();
   ctx.lineWidth = 4;
   ctx.arc(M.x,M.y,r,0,2*Math.PI);
   ctx.fillStyle = "blue";
   ctx.fill();
}

function drawText(tex,M) {
/* Einen Text anzeigen im Punkt M */
   x = M.x; y = M.y;
   ctx.font = "bold 16px Arial";
   ctx.fillStyle = "red";
   ctx.fillText(tex,x,y);
}

function autoGame() {
/* Computer-Simulation "rekursives Backtracking" starten */
  if (!neu) { alert('Bitte zuerst [New] ausführen !'); return; }
  count = 0;
  okay = false;
  i = 1, j = 1;
  backTrack(i,j);
  if (okay)  { alert('Backtracking erfolgreich beendet (n = ' + count + ')');  }
  if (!okay) { alert('Backtracking NICHT erfolgreich beendet (n = ' + count + ')'); }
  neu = false;
}

function backTrack(i,j) {
/* Computer-Simulation "rekursives Backtracking" ausführen */
  if (okay) { return; }
  if (mat[i][j] == 0) {
     drawBall(mp[i][j]);
     count++;
     mat[i][j] = 3;
     backTrack(i+1,j);
     backTrack(i+1,j+1);
     backTrack(i,j+1);
     backTrack(i-1,j+1);
     backTrack(i-1,j);
     backTrack(i-1,j-1);
     backTrack(i,j-1);
     backTrack(i+1,j-1);
  }
  if (mat[i][j] == 2) {
     drawBall(mp[i][j]);
     okay = true;
  }
}
```

```
   function changeGame() {
   /* Feld- und Farbwerte ändern */
       if (!neu) {
          alert('Bitte zuerst [New] ausführen !'); return;
       }
       change = !change;
   }

   function down(ev) {
   /* Maus drücken und Labyrinthfeld einfärben (schwarz oder weiß) */
       if (!neu) {
         alert('Bitte zuerst [New] ausführen !'); return;
       }
       x1 = 0;
       y1 = 0;
       x = ev.clientX - 60;
       y = ev.clientY - 60;
       for (i = 2; i < max; i++) {
         for (j = 2; j < max; j++) {
         pos = (Math.abs(x - mp[i][j].x) < e) && (Math.abs(y - mp[i][j].y) < e);
         if ( pos ) { x0 = mp[i][j].x; y0 = mp[i][j].y; x1 = i; y1 = j; }
         }
       }
       if (change) { mat[x1][y1] = 0; }
       else { mat[x1][y1] = 1; }
       Q = new TPoint;
       Q.x = x0; Q.y = y0;
       drawRect(Q);
//       alert(x1 + ' / ' + y1 + ' = ' +  mat[x1][y1]);
   }

   function showHelp() {
   /* Hilfetext ein-/ausblenden */
     help = !help;
     if ( help ) { document.getElementById('div1').style.visibility = 'visible'; }
     else { document.getElementById('div1').style.visibility = 'hidden'; }
   }
  </script>
</head>

<body onload = "initCanvas();">
<br>
<button id="btn1" class="cd" onclick="showHelp()"> Help </button>  
<button id="btn2" class="cd" onclick="newGame()"> New </button>  
<button id="btn3" class="cd" onclick="autoGame()"> Run </button>  
<button id="btn4" class="cd" onclick="changeGame()"> Change </button>  
<br>
<div id="div0" style="position:absolute; top:60px; left:60px">
  <canvas id='MyCanvas' width='500' height='500' onmousedown=down(event)>
  </canvas>
</div>
<br>
<p id="hp">Backtracking im Labyrinth (1) &copy Herbert Paukert</p>
<div id="div1">
  Backtracking im Labyrinth (laby1)<br>
  <br>
  Mit [New] wird ein Labyrinth mit zufälligen Hindernissen (schwarz)<br>
  erzeugt. Die freien Wegfelder sind weiß. Das mit "A" markierte Feld<br>
  ist der Eingang, das mit "Z" markierte Feld ist der Ausgang im Labyrinth.<br>
  Mit [Run] startet das Programm die Suche nach einem freien Weg von<br>
  Eingang bis Ausgang. Die Suche erfolgt mit rekursivem Backtracking.<br>
  <br>
  Hinweis 1: Wenn die Hindernisse so dicht liegen, dass kein freier<br>
  Durchgang mehr möglich ist, dann kann das Labyrinth nachträglich<br>
  mit [Change] geändert werden. Dabei werden innerhalb des Labyrinths<br>
  mit Mausklick Hindernisse entfernt oder hinzugefügt.<br>
  <br>
  Hinweis 2: Im rekursiven Backtracking startet der Computer zuerst beim<br>
  Startfeld (A) links oben. Er versucht nun die Nachbarfelder zu betreten,<br>
  beginnend mit dem direkt darunter liegenden Feld. Er wandert dann<br>
  gegen den Uhrzeigersinn in alle acht Richtungen, wenn das jeweilige<br>
  Nachbarfeld ein Hindernis (schwarz) ist. Ist das Nachbarfeld frei (weiß),<br>
  so wird das Ausgangsfeld als Spur (track) markiert und dann das<br>
  freie Nachbarfeld betreten. Dieses Verfahren wird immer wiederholt.<br>
  Trifft der Computer dabei auf ein Hindernis oder auf seine eigene Spur,<br>
  dann springt er zu jenem Feld zurück (back), wo die letzte Richtungs-<br>
  änderung stattgefunden hat. Dort wird das Verfahren dann fortgesetzt.<br>
  Dadurch wird ein Ariadne-Faden im Labyrinth simuliert. Durch Änderung<br>
  der Reihenfolge der Richtungswechsel können sich beim Backtracking<br>
  auch der Verlauf und die Länge des gefundenen Weges ändern.<br>
  <br>
  Das Verfahren ist erfolgreich beendet, wenn das Zielfeld (Z) rechts unten<br>
  erreicht wird. Das Verfahren ist dann nicht erfolgreich beendet, wenn kein<br>
  freies Feld mehr betreten werden kann.<br>
</div>
</body>
</html>
```

- **Individuelle Wegsuche im Labyrinth (laby2.html)**



**Mit [New] erzeugt der Zufallsgenerator im Spielfeld Hindernisse.
Spielfeldgröße, Hindernis-Anzahl, Wartezeit bestimmt der Spieler.
Mit [Old] kann das gleiche Spiel noch einmal wiederholt werden.
Mit [Auto] spielt der Computer mit rekursivem Backtracking alleine.
Mit [Reset] kann die HTML-Seite jederzeit neu geladen werden!**

**Mit [Go] beginnt ein Spielstein vom Anfangsfeld «A» an zu laufen.
Die Laufgeschwindigkeit ist umso größer je kleiner die Wartezeit ist.
Die Laufrichtung wird durch einen linken Mausklick auf die Schalter
[1,2,3,4,5,6,7,8,9] gesteuert, wobei der mittlere Schalter [5 = Go]
das Spiel weiter führt, wenn Spielstein und Hindernis kollidieren.
Für jede Kollision werden zur gelaufenen Weglänge fünf Strafpunkte.
addiert. Außerdem wird noch die gesamte Spielzeit gemessen.
Wird der Spielfeldrand überschritten, erfolgt immer ein Spielabbruch.**

**Das Ziel des Spiels ist es, den Spielstein auf dem kürzesten Weg und
ohne Kollisionen vom Startfeld (A) in ein Zielfeld (Z) zu befördern.
Zusätzlich sollte auch die Spielzeit möglichst kurz sein.**

**Wenn die Hindernisse zufällig so dicht liegen, dass kein freier Durch-
gang mehr möglich ist, dann kann das Labyrinth nachträglich auch
abgeändert werden. Dabei werden innerhalb des Labyrinthrandes mit
Mausklick und dann mit gehaltenener, bewegter Maustaste Hindernisse
entfernt oder hinzugefügt. Das kann mit [Change] eingestellt werden.**

**Hinweis: Die Laufrichtung kann auch mit den Ziffern 1 bis 9 auf dem
Ziffernblock der Tastatur gesteuert werden. Dazu muss jedoch vorher
die NumLock-Taste betätigt werden.**

**Hinweis: Im rekursiven Backtracking startet der Computer zuerst beim
Startfeld (A,0) links oben. Er versucht nun die Nachbarfelder zu betreten,
beginnend mit dem direkt darunter liegenden Feld. Er wandert dann
gegen den Uhrzeigersinn in alle acht Richtungen, wenn das jeweilige
Nachbarfeld ein Hindernis (schwarz,1) ist. Ist das Nachbarfeld jedoch
frei (weiß,0), so wird das Ausgangsfeld als Spur (track,3) markiert und
das freie Nachbarfeld betreten. Dieses Verfahren wird immer wiederholt.
Trifft der Computer dabei auf ein Hindernis oder auf seine eigene Spur,
dann springt er zu jenem Feld zurück (back), wo die letzte Richtungs-
änderung stattgefunden hat. Dort wird das Verfahren dann fortgesetzt.
Dadurch wird ein Ariadne-Faden im Labyrinth simulert. Das Verfahren ist
erfolgreich beendet, wenn ein Zielfeld (Z,2) erreicht wird. Es ist nicht
erfolgreich beendet, wenn kein freies Feld mehr betreten werden kann.**

**Weil beim Backtracking die Feldwerte durch die eingetragenen Spuren
verändert werden, muss nach jedem Backtracking mit [New] ein neues
Spielfeld erzeugt werden!**

```
<!doctype html>
<html>
<head>
  <title>Backtracking im Labyrinth (laby2)</title>
  <meta charset="ISO-8859-1">
  <meta name="description" content="Backtracking im Labyrinth (laby2)">
  <meta name="author" content="Herbert Paukert">

  <style>
    body  { background-color:#A0A8AF; color:black;
            font-family:Arial; font-size:16px; font-weight:bold; text-size-adjust: none;
            text-align:left; margin-left:60px; }
    .cd { border:2px solid #666666; background-color:#EEF8D8; font-size:16px; font-weight:bold;}
    .cd { border:2px solid #666666; background-color:#EEF8D8; color: darkred;
            font-size:18px; font-weight:bold; width: 50px; height: 50px; border-radius:25%;}
    #btn10 { background-color:#D8D8E8; }
    #div1 {
      position: absolute;
      left: 580px;
      top: 50px;
      width:   220px;
      height:  60px;
      margin:  10px;
      padding: 10px;
      border:  4px solid #888888;
      background: antiquewhite;
    }
    #div2 {
      position: absolute;
      left: 580px;
      top: 160px;
      width:   220px;
      height:  90px;
      margin:  10px;
      padding: 10px;
      border:  4px solid #888888;
      background: antiquewhite;
    }
    #div3 {
      position: absolute;
      left: 580px;
      top:  300px;
      width:   200px;
      height:  200px;
      margin:  10px;
      padding: 20px;
      border:  4px solid #888888;
      background: antiquewhite;
    }
    #div4 {
      position: absolute;
      left: 850px;
      top:  300px;
      width:   200px;
      height:  200px;
      margin:  10px;
      padding: 20px;
      border:  4px solid #888888;
      background: antiquewhite;
      font-size: 13px;
    }
    #p1  { position:absolute; left:600px; top:550px; }
    #hp  { position:absolute; left:60px; top:550px; }

  </style>

  <script>

   var childWindow;
   var child = false;
   window.addEventListener( "unload", function() { closeChild(); } );

   function closeChild() {
   /* Hilfetext schließen */
     if (!childWindow || childWindow.closed) { return; }
     if (child) { childWindow.document.close(); childWindow.close(); }
   }
```

```
function showHelp() {
/* Hilfetext anzeigen */
   if (child) { closeChild(); child = false; return; }
   child = true;
   childWindow = window.open('','childWindow','top=20, left=500, width=600, height=640,
       srollbars=yes, menubar=no, toolbar=no');
   childWindow.document.open();
   childWindow.document.write('<html><head></head><body style="background-color:#FFFFE8;
       font-family:Arial, sans-serif; font-size:14px; text-size-adjust:none;
       margin-left:20px; margin-bottom:40px;">');

  childWindow.document.write('<br><input type="button" value="Print" onclick="print();"><br><br>');
  childWindow.document.write('Mit dem Wechselschalter [Help] wird diese Hilfe ein-/ausgeschaltet.<br>');
  childWindow.document.write('Mit [New] erzeugt der Zufallsgenerator im Spielfeld Hindernisse.<br>');
  childWindow.document.write('Spielfeldgröße, Hindernis-Anzahl, Wartezeit bestimmt der Spieler.<br>');
  childWindow.document.write('Mit [Old] kann das gleiche Spiel noch einmal wiederholt werden.<br>');
  childWindow.document.write('Mit [Auto] spielt der Computer mit rekursivem Backtracking alleine.<br>');
  childWindow.document.write('Mit [Reset] kann die HTML-Seite jederzeit neu geladen werden!<br>');
  childWindow.document.write('<br>');
  childWindow.document.write('Mit [Go] beginnt ein Spielstein vom Anfangsfeld «A» an zu laufen.<br>');
  childWindow.document.write('Die Laufgeschwindigkeit ist umso größer je kleiner die Wartezeit ist.<br>');
  childWindow.document.write('Die Laufrichtung wird durch einen linken Mausklick auf die Schalter<br>');
  childWindow.document.write('[1, 2, 3, 4, 5, 6, 7, 8, 9] gesteuert, wobei der mittlere Schalter [5=Go]<br>');
  childWindow.document.write('das Spiel weiter führt, wenn Spielstein und Hindernis kollidieren.<br>');
  childWindow.document.write('Für jede Kollision werden zur gelaufenen Weglänge fünf Strafpunkte.<br>');
  childWindow.document.write('addiert. Außerdem wird noch die gesamte Spielzeit gemessen.<br>');
  childWindow.document.write('Wird der Spielfeldrand überschritten, erfolgt immer ein Spielabbruch.<br>');
  childWindow.document.write('<br>');
  childWindow.document.write('Das Ziel des Spiels ist es, den Spielstein auf dem kürzesten Weg<br>');
  childWindow.document.write('ohne Kollisionen vom Startfeld «A» in ein Zielfeld «Z» zu befördern.<br>');
  childWindow.document.write('Zusätzlich sollte auch die Spielzeit möglichst kurz sein.<br>');
  childWindow.document.write('<br>');
  childWindow.document.write('Wenn die Hindernisse zufällig so dicht liegen, dass kein freier Durch-<br>');
  childWindow.document.write('gang mehr möglich ist, dann kann das Labyrinth nachträglich auch<br>');
  childWindow.document.write('geändert werden. Dabei werden innerhalb des Labyrinthrandes mit<br>');
  childWindow.document.write('Mausklick und dann mit gehaltenener, bewegter Maustaste Hindernisse<br>');
  childWindow.document.write('entfernt oder hinzugefügt. Das kann mit [Change] eingestellt werden.<br>');
  childWindow.document.write('<br>');
  childWindow.document.write('Hinweis: Die Laufrichtung kann auch mit den Ziffern 1 bis 9 auf dem<br>');
  childWindow.document.write('Ziffernblock der Tastatur gesteuert werden. Dazu muss jedoch vorher<br>');
  childWindow.document.write('die NumLock-Taste betätigt werden.<br>');
  childWindow.document.write('<br>');
  childWindow.document.write('Hinweis: Im rekursiven Backtracking startet der Computer zuerst beim<br>');
  childWindow.document.write('Startfeld (A, 0) links oben. Er versucht nun die Nachbarfelder zu betreten,<br>');
  childWindow.document.write('beginnend mit dem direkt darunter liegenden Feld. Er wandert dann<br>');
  childWindow.document.write('gegen den Uhrzeigersinn in alle acht Richtungen, wenn das jeweilige<br>');
  childWindow.document.write('Nachbarfeld ein Hindernis (1) ist. Ist das Nachbarfeld jedoch frei (0),<br>');
  childWindow.document.write('so wird das Ausgangsfeld als Spur (track, 3) markiert und dann das<br>');
  childWindow.document.write('freie Nachbarfeld betreten. Dieses Verfahren wird immer wiederholt.<br>');
  childWindow.document.write('Trifft der Computer dabei auf ein Hindernis oder auf seine eigene Spur,<br>');
  childWindow.document.write('dann springt er zu jenem Feld zurück (back), wo die letzte Richtungs-<br>');
  childWindow.document.write('änderung stattgefunden hat. Dort wird das Verfahren dann fortgesetzt.<br>');
  childWindow.document.write('Dadurch wird ein Ariadne-Faden im Labyrinth simuliert.<br>');
  childWindow.document.write('<br>');
  childWindow.document.write('Das Verfahren ist erfolgreich beendet, wenn ein Zielfeld (Z, 2) erreicht wird.<br>');
  childWindow.document.write('Das Verfahren ist nicht erfolgreich beendet, wenn kein freies Feld mehr<br>');
  childWindow.document.write('betreten werden kann.<br>');
  childWindow.document.write('<br>');
  childWindow.document.write('Weil beim Backtracking die Feldwerte durch die eingetragenen Spuren<br>');
  childWindow.document.write('verändert werden, muss nach jedem Backtracking mit [New] ein neues<br>');
  childWindow.document.write('Spielfeld erzeugt werden!<br>');

  childWindow.document.write('</body></html>');
 }

 /* globale Variable (Anfang) ------------------------------------------ */

  var canvas;              //  grafische Leinwand (Spielfeld)
  var ctx;                 //  Kennzeichen der Leinwand

  var len = 500;           //  Abmessungen des quadratischen Rasters
  var min = 10;            //  Minimale Rasterzeilen (bzw. Spalten)
  var max = 16;            //  Maximale Rasterzeilen (bzw. Spalten)
  var d = len / max;       //  Größe eines Rasterfeldes
  var e = d / 2;           //  Halbe Länge eines Rasterfeldes
  var anz = 30;            //  Anzahl der Hindernisse
  var zeit = 1300;         //  Wartezeit bei der Bewegung
  var play;                //  Timervariable für die Bewegung
  var numlock = false;     //  Wechselschalter für "NumLock"

  var neu = false;         //  Neues Spielfeld
  var old = false;         //  Altes Spielfeld (Kopie des neuen Spielfeldes)
  var start = true;        //  Start des Spieles
  var go = false;          //  Start der Bewegungen
  var end = false;         //  Ende des Spiels
  var auto = false;        //  Steuervariable für das automatische Backtracking
  var count = 0;           //  Schrittzähler beim Backtracking

  var color = false;       //  Wechselschalter für Ballfarbe
  var richt = 0;           //  Bewegungsrichtungen (1 bis 9)
  var weg   = 0;           //  Länge des zurückgelegten Weges
  var kolis = 0;           //  Anzahl der Kollisionen
  var score = 0;           //  Erreichter Punktewert (= weg + 5*kolis)
```

```
var starttime = 0;       //  Startzeit
var endtime = 0;         //  Endzeit
var finaltime = 0;       //  Umgerechnete Endzeit

var dx = 0;              //  Schrittweite in x-Richtung
var dy = 0;              //  Schrittweite in y-Richtung
var x0 = 0;              //  x-Wert des Anfangspunktes
var y0 = 0;              //  y-Wert des Anfangspunktes
var change = false;      //  Wechselschalter (schwarz - weiß)
var doChange = false;    //  Wechselschalter für Spielfeldänderungen mit Maus

var mp = new Array();    // Mittelpunkte der Rasterfelder
for (i = 0; i <= max; i++) { mp[i] = new Array(max); }

var mat = new Array();   // Neuer Spielfeldraster
for (i = 0; i <= max; i++) { mat[i] = new Array(max); }

var cop = new Array();   // Kopie des Spielfeldraster
for (i = 0; i <= max; i++) { cop[i] = new Array(max); }

function TPoint(tx,ty)   // Punkt im Spielfeld
{
  this.x = tx;
  this.y = ty;
}
var P = new TPoint;
var Q = new TPoint;

/* globale Variable (Ende) ------------------------------------------- */

function myTrim(x) { return x.replace(/^\s+|\s+$/gm,'') }
function startclock() {var today = new Date(); starttime = today.getTime();}
function endclock()   {var today = new Date(); endtime = today.getTime();}
function calctime()   {var time = (endtime - starttime)/1000; return time;}

function fmove(i) {
/* Ballrichtung bestimmen */
  if (i == 5) { runGame(); }
  else { richt = i; }
}

function KeyListener(ev) {
/* Tastatur-Event-Handler */
  var taste = ev.which || ev.keycode;
/* alert(taste); */
  if (taste == 144) {
     numlock = !numlock;
     if (numlock) {
        document.getElementById('p1').innerHTML = ' «NumLock-Tasten» EIN ' +
                                                  '<br>' + '1, 2, 3, 4, 5, 6, 7, 8, 9';
     }
     else { document.getElementById('p1').innerHTML = ' «NumLock-Tasten» AUS '; }
  }
  if (taste == 103) { fmove(1); }
  if (taste == 104) { fmove(2); }
  if (taste == 105) { fmove(3); }
  if (taste == 100) { fmove(4); }
  if (taste == 101) { fmove(5); }
  if (taste == 102) { fmove(6); }
  if (taste == 97)  { fmove(7); }
  if (taste == 98)  { fmove(8); }
  if (taste == 99)  { fmove(9); }
}

function drawM() {
/* Mittelpunkte der Rasterfelder bestimmen */
  for (i = 1; i <= max; i++) {
    for (j = 1; j <= max; j++) {
        a = (i-1)*d + e; b = (j-1)*d + e; mp[i][j] = new TPoint(b,a);
        mat[i][j] = 0;
    }
  }
}

function initMatrix(anz) {
/* Matrix initialisieren */
  for (i = 1; i <= max; i++) {
    for (j = 1; j <= max; j++) {
        mat[i][j] = 0; cop[i][j] = 0;
    }
  }
```

```
  for (i = 1; i <= max; i++) {
      mat[i][1] = 1;
      mat[i][max] = 1;
  }
  for (j = 1; j <= max; j++) {
      mat[1][j] = 1;
      mat[max][j] = 1;
  }

  p = 0;
  n = max * max - 40;
  while (p < anz ) {
    if (p > n ) { anz = n; break; }
    i = Math.floor(max * Math.random() ) + 1;
    j = Math.floor(max * Math.random() ) +1;
    if (mat[i][j] != 1)  {
       p = p + 1;
       mat[i][j] = 1;
    }
  }

  for (i = 1; i <= 3; i++) { mat[i][1] = 0; }
  for (i = 1; i <= 3; i++) { mat[max-i][max] = 2; }
  mat[1][2] = 0;
  mat[1][3] = 0;
  mat[2][2] = 0;
  for (i = 1; i <= max; i++) {
    for (j = 1; j <= max; j++) {
        cop[i][j] = mat[i][j];
    }
  }
}

function copyMatrix(max) {
/* Matrix zurück kopieren */
  for (i = 1; i <= max; i++) {
    for (j = 1; j <= max; j++) {
        mat[i][j] = cop[i][j];
    }
  }
}

function showMatrix(max) {
/* Matrix anzeigen */
  change = false;
  for (i = 1; i <= max; i++) {
    for (j = 1; j <= max; j++) {
         z = mat[i][j];
         if (z == 1) { drawRect(mp[i][j]); }
         drawText(' ',mp[i][j]);
    }
  }
  for (i = 1; i <= 3; i++) { drawText('A',mp[i][1]); }
  for (i = 1; i <= 3; i++) { drawText('Z',mp[max-i][max]); }
  drawBorder();
}

function startInit() {
/* Spielvariable initialisieren */
   start = true;
   weg = 0;
   kolis = 0;
   score = 0;
   starttime = 0;
   endtime = 0;
   finaltime = 0;
   document.getElementById('s3').innerHTML = 'Weglänge = ' + weg;
   document.getElementById('s4').innerHTML = 'Kollisionen = ' + kolis;
   document.getElementById('s5').innerHTML = 'Punktewert = ' + score;
   document.getElementById('s6').innerHTML = 'Spielzeit = ' + finaltime;
}

function newGame() {
/* Neues Spiel initialisieren */
  if (go) { return; }
  startInit();
  initCanvas();
  neu = true;
  old = true;
  end = false;
```

```
  zahl = min;
  zahl = prompt('Größe des Spielfeldes (10 - 16)',zahl);
  zahl = myTrim(zahl);
  if (!isNaN(zahl))
  {
    if ((zahl < 10) || ( zahl > 16)) { zahl = 12; }
  }
  else { zahl = 12; }
  max = zahl;
  min = zahl;

  zahl = anz;
  zahl = prompt('Anzahl der Hindernisse (20 - 90)',zahl);
  zahl = myTrim(zahl);
  if (!isNaN(zahl))
  {
    if ((zahl < 20) || ( zahl > 90)) { zahl = 30; }
  }
  else { zahl = 30; }
  anz = zahl;

  zahl = zeit;
  zahl = prompt('Wartezeit in Millisekunden (800 - 1800)',zahl);
  zahl = myTrim(zahl);
  if (!isNaN(zahl))
  {
    if ((zahl < 800) || ( zahl > 1800)) { zahl = 1300; }
  }
  else { zahl = 1300; }
  zeit = zahl;
  d = len / max;
  e = d / 2;
  document.getElementById('s0').innerHTML = ' Spielgröße = ' + max;
  document.getElementById('s1').innerHTML = ' Hindernisse = ' + anz;
  document.getElementById('s2').innerHTML = ' Wartezeit = ' + zeit + ' MSec';
  drawField();
  drawM();
  initMatrix(anz);
  showMatrix(max);
  weg   = 0;
  kolis = 0;
  score = 0;
  P.x = 1;
  P.y = 1;
  Q = mp[P.y][P.x];
  drawBall(Q);
}

function oldGame() {
/* Altes Spiel wiederholen */
  if (go) { return; }
  if ( (!neu) && (!old) ) {
     alert('Bitte zuerst [New] ausführen !');
     return;
  }
  old = true;
  neu = true;
  startInit();
  end = false;
  document.getElementById('s0').innerHTML = ' Spielgröße = ' + max;
  document.getElementById('s1').innerHTML = ' Hindernisse = ' + anz;
  document.getElementById('s2').innerHTML = ' Wartezeit = ' + zeit + ' MSec';
  initCanvas();
  drawField();
  drawM();
  copyMatrix(max);
  alert('Spielfeld wird restauriert');
  showMatrix(max);
  weg   = 0;
  kolis = 0;
  score = 0;
  P.x = 1;
  P.y = 1;
  Q = mp[P.y][P.x];
  drawBall(Q);
}
```

```
function initCanvas() {
/* Canvas initialisieren */
   canvas = document.getElementById('MyCanvas');
   ctx = canvas.getContext('2d');
   ctx.lineWidth = 4;
   ctx.fillStyle = "white";
   ctx.fillRect(0,0,len,len);
   ctx.strokeStyle = "black";
   ctx.strokeRect(0,0,len,len);
   ctx.lineWidth = 2;
}

function drawField() {
/* Raster zeichnen */
   ctx.lineWidth = 1;
   ctx.strokeStyle = "black";
   ctx.beginPath();
   for (i = 0; i < max; i++) {
     ctx.moveTo(0,i*d);
     ctx.lineTo(len,i*d);
   }

   for (i = 0; i < max; i++) {
     ctx.moveTo(i*d,0);
     ctx.lineTo(i*d,len);
   }

   ctx.stroke();
}

function drawBorder() {
/* Rahmen zeichnen */
   ctx.lineWidth = 4;
   ctx.strokeStyle = "black";
   ctx.strokeRect(0,0,500,500);
}

function drawRect(M) {
/* Ein Rasterquadrat mit Mittelpunkt M zeichnen und einfärben */
   r = e-1
   var l = (M.x - r);
   var t = (M.y - r);
   var w = 2*r;
   var h = 2*r;
   if (!change) { ctx.fillStyle = "rgb(80,80,80)"; }
   else { ctx.fillStyle = "rgb(255,255,255)"; }
   ctx.fillRect(l,t,w,h);
}

function drawBall(M) {
/* Einen Kreis mit Mittelpunkt M zeichnen und einfärben */
   r = 8;
   ctx.beginPath();
   ctx.lineWidth = 4;
   ctx.strokeStyle = "yellow";
   ctx.arc(M.x,M.y,r,0,2*Math.PI);
   ctx.stroke();

   if (!auto) {
      if (!color) { ctx.fillStyle = "red"; }
      else { ctx.fillStyle = "darkred"; }
   }

   if (auto) { ctx.fillStyle = "blue"; }
   ctx.fill();
}

function drawText(tex,M) {
/* Einen Text im Punkt M anzeigen */
   x = M.x;
   y = M.y;
   ctx.font = "bold 16px Arial";
   ctx.fillStyle = "red";
   ctx.fillText(tex,x,y);
}
```

```
function goTO() {
/* Zeitgesteuertes Spiel ausführen */
  dx = 0;
  dy = 1;
  if (richt == 1) { dx = -1; dy = -1; }
  if (richt == 2) { dx =  0; dy = -1; }
  if (richt == 3) { dx =  1; dy = -1; }
  if (richt == 4) { dx = -1; dy =  0; }
  if (richt == 6) { dx =  1; dy =  0; }
  if (richt == 7) { dx = -1; dy =  1; }
  if (richt == 8) { dx =  0; dy =  1; }
  if (richt == 9) { dx =  1; dy =  1; }
  P.x = P.x + dx;
  P.y = P.y + dy;
  if ( (P.x >= 1) && (P.x <= max) && (P.y >= 1) && (P.y <= max) ) {
    Q = mp[P.y][P.x];
    drawBall(Q);
    weg = weg + 1;
    if ( mat[P.y][P.x] == 1 ) {
       kolis = kolis + 1;
       score = 1*weg + 5*kolis;
       color = !color;
       clearInterval(play);
       document.getElementById('s3').innerHTML = 'Weglänge = ' + weg;
       document.getElementById('s4').innerHTML = 'Kollisionen = ' + kolis;
       alert('Kollision - Richtung wählen und [GO]');
       go = false;
    }
    if ( mat[P.y][P.x] == 2 ) {
       weg = weg + 1;
       score = 1*weg + 5*kolis;
       clearInterval(play);
       endclock();
       finaltime = calctime();
       document.getElementById('s3').innerHTML = 'Weglänge = ' + weg;
       document.getElementById('s4').innerHTML = 'Kollisionen = ' + kolis;
       document.getElementById('s5').innerHTML = 'Punktewert = ' + score;
       document.getElementById('s6').innerHTML = 'Spielzeit = ' + finaltime;
       alert('Ziel erreicht - Spielzeit = ' + finaltime + ' Sekunden. ');
       P.x = 1; P.y = 1;
       start = false;
       go = false;
       end = true;
    }
  }
  else {
     clearInterval(play);
     alert('Spielfeldrand überschritten - Spielabbruch !');
     refresh();
  }
}

function runGame() {
/* Spiel starten */
   if (!neu) { alert('Bitte zuerst [New] oder [Old] ausführen !'); return; }
   if (end) { return; }
   if (go) { return; }
   go = true;
   if (start) { startclock(); }
   start = false;
   play = setInterval(goTO,zeit);
}

function autoGame() {
/* Computer-Simulation "rekursives Backtracking" starten */
  if (go) { return; }
  if (!neu) { alert('Bitte zuerst [New] oder [Old] ausführen !'); return;  }
  auto = true;
  startInit();
  mat[1][1] = 0; mat[2][1] = 0; mat[3][1] = 0; mat[1][2] = 0;
  count = 0;
  i = 1; j = 1;
  backTrack(i,j);
  if (!auto) { alert('Backtracking erfolgreich beendet (n = ' + count + ')'); }
  if (auto)  { alert('Backtracking NICHT erfolgreich beendet (n = ' + count + ')'); }
  auto = false;
  neu = false;
}
```

```
    function backTrack(i,j) {
    /* Computer-Simulation "rekursives Backtracking" ausführen */
      if (!auto) { return }
      if (mat[i][j] == 0) {
         drawBall(mp[i][j]);
         count++;
         mat[i][j] = 3;
         backTrack(i+1,j);
         backTrack(i+1,j+1);
         backTrack(i,j+1);
         backTrack(i-1,j+1);
         backTrack(i-1,j);
         backTrack(i-1,j-1);
         backTrack(i,j-1);
         backTrack(i+1,j-1);
      }
      if (mat[i][j] == 2 ) {
         drawBall(mp[i][j]);
         auto = false;
      }
   }

   function refresh() {
   /* die Internetseite neu laden */
        window.location.reload();
   }

   function changeGame() {
   /* Feld- und Farbwerte ändern */
        if (go) { return; }
         if (!neu) {
          alert('Bitte zuerst [New] oder [Old] ausführen !');
          return;
        }
        change = !change;
   }

   function move(ev) {
   /* Maus bewegen */
       if (!doChange) { return; }
       x1 = 0;
       y1 = 0;
       x = ev.clientX - 60;
       y = ev.clientY - 60;
       for (i = 2; i < max; i++) {
         for (j = 2; j < max; j++) {
         pos = (Math.abs(x - mp[i][j].x) < e) && (Math.abs(y - mp[i][j].y) < e);
         if ( pos ) { x0 = mp[i][j].x; y0 = mp[i][j].y; x1 = i; y1 = j; }
         }
       }
       if (!change) { mat[x1][y1] = 1; cop[x1][y1] = 1; }
       else { mat[x1][y1] = 0; cop[x1][y1] = 0; }
       var R = new TPoint;
       R.x = x0; R.y = y0;
       drawRect(R);
   }

   function down(ev) {
   /* Maus drücken */
       if (go) { return; }
       if (!neu) {
         alert('Bitte zuerst [New] ausführen !');
         return;
       }
       doChange = true;
   }

   function up(ev) {
   /* Maus loslassen */
       doChange = false;
   }

  </script>
</head>
```

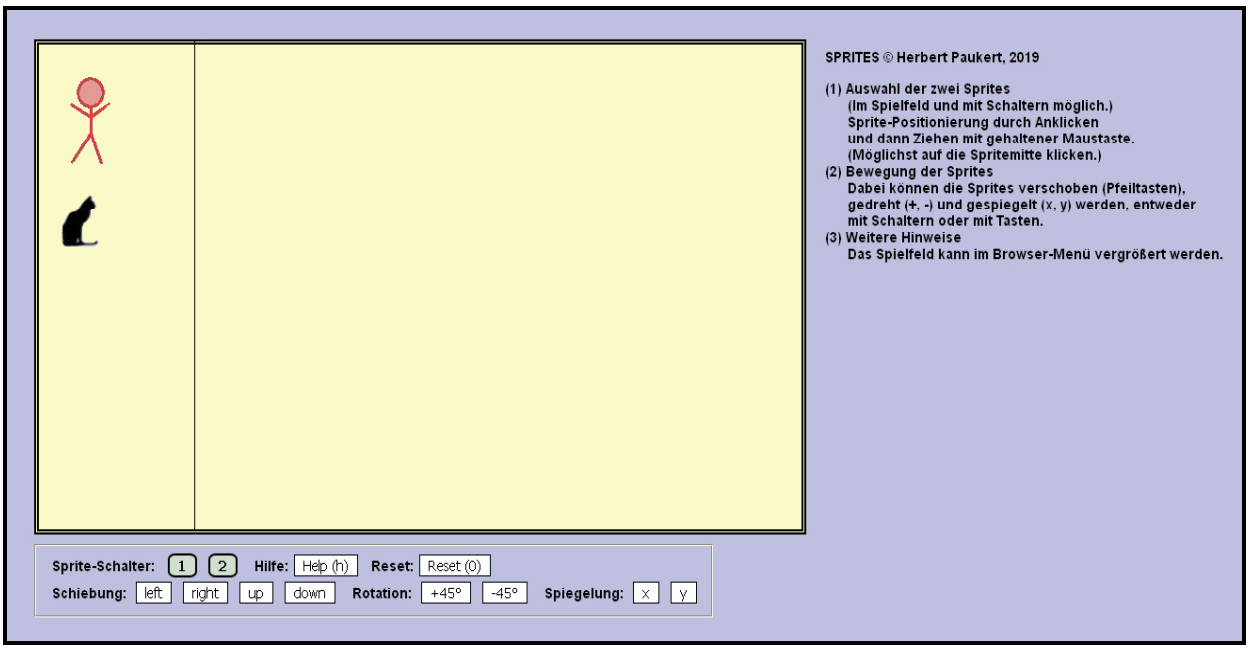
```
<body onload = "initCanvas();" onkeydown= "KeyListener(event);" >
<br>
<button id="btn1" class="cd" onclick="showHelp()"> Help </button>  
<button id="btn2" class="cd" onclick="newGame()"> New </button>  
<button id="btn3" class="cd" onclick="oldGame()"> Old </button>  
<button id="btn4" class="cd" onclick="autoGame()"> Auto </button>  
<button id="btn5" class="cd" onclick="runGame()"> GO </button>  
<button id="btn16" class="cd" onclick="changeGame()"> Change </button>  
<button id="btn15" class="cd" onclick="refresh()"> Reset </button>
<br>

<div id="div0" style="position:absolute; top:60px; left:60px">
  <canvas id='MyCanvas' width='500' height='500'
    onmousedown=down(event) onmouseup=up(event) onmousemove=move(event)>
  </canvas>
</div>
<br>

<div id="div1">
<span id="s0">Spielgröße = 0</span><br>
<span id="s1">Hindernisse = 0</span><br>
<span id="s2">Wartezeit = 0</span>
</div>

<div id="div2">
<span id="s3">Weglänge = 0</span><br>
<span id="s4">Kollisionen = 0</span><br>
<span id="s5">Punktewert = 0</span><br>
<span id="s6">Spielzeit = 0</span>
</div>

<div id="div3">
   <button id="btn6"  class="cd1" onclick="fmove(1)"> 7 </button>  
<button id="btn7"  class="cd1" onclick="fmove(2)"> 8 </button>  
<button id="btn8"  class="cd1" onclick="fmove(3)"> 9 </button><br><br>
   <button id="btn9"  class="cd1" onclick="fmove(4)"> 4 </button>  
<button id="btn10" class="cd1" onclick="fmove(5)"> GO </button>  
<button id="btn11" class="cd1" onclick="fmove(6)"> 6 </button><br><br>
   <button id="btn12" class="cd1" onclick="fmove(7)"> 1 </button>  
<button id="btn13" class="cd1" onclick="fmove(8)"> 2 </button>  
<button id="btn14" class="cd1" onclick="fmove(9)"> 3 </button>
</div>

<div id="div4">
Richtungssteuerung mit Maus:<br>
(immer <font color = "red"><u>vor GO</u></font> anklicken)<br>
<br>
7 = links hinauf<br>
8 = hinauf<br>
9 = rechts hinauf<br>
4 = links<br>
5 = GO<br>
6 = rechts<br>
1 = links hinunter<br>
2 = hinunter<br>
3 = rechts hinunter<br>
</div>

<p id='p1'> «NumLock-Tasten»</p>
<p id="hp">Backtracking im Labyrinth (2) &copy; Herbert Paukert</p>
</body>
</html>
```

[6.1.5] Zwei Spriteprogramme

- **Sprites moving (sprites.html)**

Hinweis: Das vorliegende Programm soll einen Überblick über die Verwendung von Sprites liefern. Sprites sind nichts anderes als kleine rechteckige Bilder, die mit einem **transparenten Hintergrund** ausgestattet sind, und welche mit Maus oder Tasten in einem Spielfeld bewegt werden können.

Erwähnenswert ist die Eigenschaft „**user-select**“. Diese bestimmt, ob der Text eines Objektes mit der Maus ausgewählt werden kann. Standardmäßig ist der Wert auf „**all**“ eingestellt, was bei einem Doppelklick zu einer hervorgehobenen Markierung des Textes führt, der dann auch verschoben werden kann. Wird der Wert auf „**None**“ gesetzt, dann kann kein Text ausgewählt werden, was im vorliegenden Programm bereits im „body“ der Fall ist.




```
<!DOCTYPE html">
<html>
<head>
<title>sprites.html </title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=0.75, user-scalable=no">
<meta name="description" content="Sprite-Spielprogramm">
<meta name="author" content="Herbert Paukert">

<style>

body  { background: #BFBFDF; color: black; font-family: sans-serif; font-size: 18px;
       -ms-user-select: None; -moz-user-select: None; -webkit-user-select: None; user-select: None; }
.btn2 { border: 2px solid #000000; border-radius: 25%; color: black; background: #D0DFD0;
        font-size: 26px; font-weight: bold; margin: 2px; }
.btn1 { border: 1px solid #000000; color: black; background: #FFFFFF;
        font-size: 26px; margin: 2px; }

#sprite1 {
  position: absolute;
  left:   20px;
  top:    20px;
  width:  70px;
  height:100px;
  background-image: url('man.png');
}

#sprite2 {
  position: absolute;
  left:   20px;
  top:    140px;
  width:  50px;
  height: 60px;
  background-image: url('cat.png');
}
```

```
#surface {
  position: absolute;
  background-color: rgb(250,250,200);
  border: black 5px double;
  left:    10px;
  top:     10px;
  width:  810px;
  height: 470px;
}

#start {
  position: absolute;
  border-right: black 1px solid;
  left:    10px;
  top:     10px;
  width:  170px;
  height: 475px;
}

#controls {
  position: absolute;
  top: 500px;
}
#fs { width: 860px; }
</style>

<script>
   var childWindow;
   var child = false;
   window.addEventListener( "unload", function() { closeChild(); } );

   function closeChild() {
   /* Hilfetext schließen */
     if (!childWindow || childWindow.closed) { return; }
     if (child) { childWindow.document.close(); childWindow.close(); }
   }
   function showHelp() {
   /* Hilfetext anzeigen */
     if (child) { closeChild(); child = false; return; }
     child = true;
     childWindow = window.open('','childWindow','top=25, left=850, width=450, height=550, srollbars=yes, menubar=no,
                               toolbar=no');
     childWindow.document.open();
     childWindow.document.write('<html><head><meta name="viewport" content="width=device-width, initial-
                                 scale=1.0"></head>');
     childWindow.document.write('<body style="background-color:#F8F8F8; font-family: sans-serif; font-size:14px;
                                text-size-adjust:none; padding-left:10px; padding-bottom:10px;">');
     childWindow.document.write('<br><input type="button" value="Print" onclick="print();"><br><br>');
     childWindow.document.write('SPRITES-MOVE-DEMO, Version 5.0<br><br>');
     childWindow.document.write('(1) Auswahl der Sprites:<br>');
     childWindow.document.write('Im Spielfeld und mit Schaltern möglich.<br>');
     childWindow.document.write('Möglichst auf die Spritemitte klicken.<br><br>');
     childWindow.document.write('(1a) Figuren-Positionierung mittels Ziehen<br>');
     childWindow.document.write('und Loslassen mit der Maus (Maus-Move).<br>');
     childWindow.document.write('Automatisch bei Desktops mit Widescreens.<br><br>');
     childWindow.document.write('(1b) Figuren-Positionierung mittels Klick-Klick:<br>');
     childWindow.document.write('Erstens durch Anklicken des Figuren-Schalters.<br>');
     childWindow.document.write('Zweitens durch Anklicken der Figuren-Position.<br>');
     childWindow.document.write('Automatisch bei Phones mit Smallscreens.<br><br>');
     childWindow.document.write('(2) Bewegung der Sprites:<br>');
     childWindow.document.write('Dabei können die Sprites verschoben<br>');
     childWindow.document.write('(Pfeiltasten), gedreht (+, -) und auch<br>');
     childWindow.document.write('gespiegelt (x, y) werden, entweder mit<br>');
     childWindow.document.write('Schaltern oder mit Tasten.<br><br>');
     childWindow.document.write('<b>[Klick on/off]</b> wechselt jederzeit zwischen<br>');
     childWindow.document.write('Maus-Move-Technik und Klick-Klick-Technik.<br><br>');
     childWindow.document.write('Hinweis zum Optimieren auf Smartphones:<br>');
     childWindow.document.write('Ev. Verwendung eines Berührungsstiftes!<br><br>');
     childWindow.document.write('(c) Herbert Paukert<br><br>');
     childWindow.document.write('</body></html>');
   }

// -------------------- globale Variable (Beginn) --------------------
var MIN_X =   0;                // Spielfeldgrenzen
var MAX_X = 760;
var MIN_Y =   0;
var MAX_Y = 380;
var MAX_S = 100;                // Koordinaten-Ausgleich für das Spielfeld
var dxy = 15;                   // Koordinaten-Ausgleich für die Spielfiguren

var sprite;                     // Sprite-Objekt
var arr = [];                   // Array der Sprites
arr[0] = 0;
var num = 0;                    // Sprite-Nummer
var znum = 0;                   // z-Index des aktuellen Sprites
var anz = 2;                    // Anzahl der Sprite-Objekte
var lSprite = new Array(anz+1);  // Sprite-Left
var tSprite = new Array(anz+1);  // Sprite-Top
var wSprite = new Array(anz+1);  // Sprite-Width
var hSprite = new Array(anz+1);  // Sprite-Height

var smallScreen = 640;          // Screen von Smartphones
var desktop = true;             // Spielvariante entweder mit Mausziehen oder Mausklicken
var run = false;                // Spielvariable
```

```
var wied = false;               // Kennvariable für Zugsbeginn (bei Klick-Klick-Technik)
var neu = false;                // Kennvariable für neue Sprite-Figur (bei Klick-Klick-Technik)
var neunum = 0;                 // Neue Sprite-Nummer (bei Klick-Klick-Technik)

var x = 10;                     // x-Koordinate eines Punkes im Spielfeld
var y = 10;                     // y-Koordinate eines Punkes im Spielfeld

var step = 1;                   // Schiebungslänge
var rwin = 45;                  // Rotationswinkel
var rotold = 0;                 // Gespeicherter Rotationswinkel
var mirdir = 0;                 // Gespeicherte Spiegelungsrichtung
var mirold = 1;                 // Wechselschalter für Spiegelungen (an x- oder y-Achse)

var helpFlag = false;           // Wechselschalter für Hilfstext
// -------------------- globale Variable (Ende) --------------------

// Left, Top, Width und Height der Sprites initialisieren
for (i = 1; i <= anz; i++) {
   lSprite[i] = 0;
   tSprite[i] = 0;
   wSprite[i] = 0;
   hSprite[i] = 0;
}

function init(){
// Sprites initialisieren
  if (screen.width <= smallScreen) { desktop = false; }
  if (screen.width > smallScreen)  {
     desktop = true;
     var x = document.getElementsByClassName("btn1");
     var y = document.getElementsByClassName("btn2");
     var i;
     for (i = 0; i < x.length; i++) { x[i].style.fontSize = "18px"; }
     for (i = 0; i < y.length; i++) { y[i].style.fontSize = "18px"; }
     document.getElementById("fs").style.width = "790px";
  }
  for (i=1; i <= anz; i++) {
    id = 'sprite' + i;
    el = document.getElementById(id);
    arr[i] = el;
    lSprite[i] = parseInt(el.style.left);
    tSprite[i] = parseInt(el.style.top);
    wSprite[i] = parseInt(el.style.width);
    hSprite[i] = parseInt(el.style.height);
//   alert(id + ':' + '\n' + lSprite[i] + ' / ' + tSprite[i] + '\n' + wSprite[i] + ' / ' + hSprite[i]);
  }
  document.onkeydown   = keyListener;
  document.onmousedown = mDown;
  document.onmouseup   = mUp;
  document.onmousemove = mMove;
}

function checkBounds(x,y) {
// Spiefeldrand prüfen für die Mouse-Move-Technik
  if ( (x < MIN_X) || (x > MAX_X) || (y < MIN_Y) || (y > MAX_Y) ) { run = false; return false; }
  else {return true; }
}

function setBounds(x,y) {
// Spiefeld begrenzen für die Klick-Klick-Technik
  if (x < MIN_X){ x = MIN_X; return false; }
  if (x > MAX_X){ x = MAX_X; return false; }
  if (y < MIN_Y){ y = MIN_Y; return false; }
  if (y > MAX_Y){ y = MAX_Y; return false; }
  return true;
}

function checkPlay(x,y) {
// Test, ob Mauscursor im Spielfeld ist
    a = MIN_X;
    b = MIN_Y;
    e = MAX_X + MAX_S;
    f = MAX_Y + MAX_S;
    if ( (x >= a) && (x <= e) && (y >= b) && (y <= f) ) { return true; }
    else { run = false; return false; }
}

function getSprite(x,y) {
// Sprite suchen und Spritenummer ermitteln
  var n = 0;
  for (i = 1; i <= anz; i++) {
    a = lSprite[i];
    b = tSprite[i];
    c = wSprite[i];
    d = hSprite[i];
    e = 1*a + 1*c + dxy;
    f = 1*b + 1*d + dxy;
    if ( (x >= a) && (x <= e) && (y >= b) && (y <= f) ) { n = i; break; }
  }
  return n;
}
```

```
function mDown(event) {
// Sprite-Auswahl mit "mousedown"
if (desktop) {
  run = true;
  x = event.pageX;
  y = event.pageY;
  okay = checkPlay(x,y);
  if (!okay) { run = false; return; }
  num = getSprite(x,y);
  sprite = arr[num];
  znum += 1;
  sprite.style.zIndex = znum;      // stellt die Figur in den Vordergrund
  return;
}
// nur aktiv bei Klick-Klick-Technik
if (!wied) {
  run = true;
  x = event.pageX;
  y = event.pageY;
  innen = checkBounds(x,y);
  if (!innen) { return; }
  okay = checkPlay(x,y);
  if (!okay) { return; }
  num = getSprite(x,y);
  if (neu) { num = neunum; }
  sprite = arr[num];
  znum += 1;
  sprite.style.zIndex = znum;      // stellt die Figur in den Vordergrund
  wied = true;
  return;
}
// nur aktiv bei Klick-Klick-Technik
if ( (wied) || (neu) ) {
  if (!run) {return; }
  height = parseInt(sprite.style.height);
  width  = parseInt(sprite.style.width);
  x = event.pageX - (width/2);
  y = event.pageY - (height/2);
  okay = setBounds(x,y);
  if (!okay) { return; }
  lSprite[num] = x;
  tSprite[num] = y;
  sprite.style.left = x + "px";
  sprite.style.top = y + "px";
  s = 'scale(1)';
}
}

function mUp(event) {
// Sprite-Bewegung beenden mit "mouseup"
  if (!desktop) { return; }
  run = false;
}

function mMove(event) {
// Sprite-Bewegung ausführen mit "mousemove"
  if (!desktop) { return; }
  if (!run) { return; }
  height = parseInt(sprite.style.height);
  width  = parseInt(sprite.style.width);
  x = event.pageX - (width/2) - dxy;
  y = event.pageY - (height/2) - dxy;
  innen = checkBounds(x,y);
  if (!innen) { run = false; return; }
  lSprite[num] = x;
  tSprite[num] = y;
  sprite.style.left = x + "px";
  sprite.style.top = y + "px";
  s = 'scale(1)';
  if (x < 140) {
    sprite.style.transform = s;
    sprite.style.webkitTransform = s;
  }
}

function keyListener(event){
// Tastatur-Handler
// alert(event.keyCode);
  if (event.keyCode == 48)  { refresh(); }
  if (event.keyCode == 49)  { num = 1; sprite = arr[num]; }
  if (event.keyCode == 50)  { num = 2; sprite = arr[num]; }
  if (event.keyCode == 37)  { moveSprite(-step, 0); }    // left
  if (event.keyCode == 38)  { moveSprite(0, -step); }    // up
  if (event.keyCode == 39)  { moveSprite(step, 0);  }    // right
  if (event.keyCode == 40)  { moveSprite(0, step);  }    // down
  if (event.keyCode == 171) { rotateSprite(+rwin); }     // rotation +
  if (event.keyCode == 173) { rotateSprite(-rwin); }     // rotation -
  if (event.keyCode == 88)  { mirrorSprite(0); }         // mirror X
  if (event.keyCode == 89)  { mirrorSprite(1); }         // mirror Y
  if (event.keyCode == 83)  { select(); }                // select
  if (event.keyCode == 76)  { solve();  }                // solve
  if (event.keyCode == 72)  { showHelp(); }              // help
}
```

```
function moveSprite(dx, dy){
// Sprite verschieben
  x = parseInt(sprite.style.left);
  y = parseInt(sprite.style.top);
  x += dx;
  y += dy;
  okay = checkBounds(x,y);
  if (!okay) { return; }
  lSprite[num] = x;
  tSprite[num] = y;
  sprite.style.left = x + "px";
  sprite.style.top = y + "px";
  wied = false;
}
function rotateSprite(w) {
// Sprite rotieren (um den Sprite-Mittelpunkt)
  w = rotold + w;
  s = '';
  if ( mirdir == 0) { s = 'rotate(' + w + 'deg)' + ' scaleX(' + mirold + ')' ; }
  if ( mirdir == 1) { s = 'rotate(' + w + 'deg)' + ' scaleY(' + mirold + ')' ; }
  rotold = w;
  sprite.style.transformOrigin = "50% 50%";
  sprite.style.webkitTransformOrigin = "50% 50%";
  sprite.style.transform = s;
  sprite.style.webkitTransform = s;
  wied = false;
}
function mirrorSprite(r) {
// Sprite spiegeln (an X- oder Y-Achse)
  mirdir = r;
  mirror = (-1) * mirold;
  s = '';
  if ( mirdir == 0) { s = ' scaleX(' + mirror + ')' +  ' rotate(' + rotold + 'deg)'; }
  if ( mirdir == 1) { s = ' scaleY(' + mirror + ')' +  ' rotate(' + rotold + 'deg)'; }
  mirold = mirror;
  sprite.style.transformOrigin = "50% 50%";
  sprite.style.webkitTransformOrigin = "50% 50%";
  sprite.style.transform = s;
  sprite.style.webkitTransform = s;
  wied = false;
}
function clickSprite(node) {
// Zusätzliche Sprite-Auswahl mittels Zahlen-Schalter (Klick-Klick-Technik)
  s = node.id;
  t = s.substr(2,1)
  num = parseInt(t);
  sprite = arr[num];
  neunum = num;
  neu = true;
  wied = false;
}
function refresh() {window.location.reload();}
function clickDesk() {
// Klick-Move-Toggle (Wechsel zwischen Mouse-Move und Klick-Klick)
   desktop = !desktop;
   if (desktop)  { alert('Positionieren mit Maus-Move !'); }
   if (!desktop) { alert('Positionieren mit Klick-Klick !'); }
}
</script>
</head>

<body onload="init();">
  <div id = "surface">
    <div id="sprite1" style="position:absolute; left:20px; top:20px; width:70px; height:100px;"></div>
    <div id="sprite2" style="position:absolute; left:20px; top:140px; width:50px; height:60px;"></div>
  </div>
  <div id = "start"> </div>
  <form action = "" id = "controls">
        <fieldset id="fs">
          Sprites:  
        <button type = "button" class="btn2" id="bn1" onclick="clickSprite(this)">1</button> 
        <button type = "button" class="btn2" id="bn2" onclick="clickSprite(this)">2</button> 
         Klick: <button type = "button" class="btn1" onclick="clickDesk()"> on/off
        </button>   Hilfe: <button type = "button" class="btn1" onclick="showHelp()">
        on/off </button>   Neustart: <button type = "button" class="btn1"
        onclick="refresh()"> Reset </button> 
        <br>
          Schieben: 
        <button type = "button" class="btn1" onclick = "moveSprite(-step, 0)"> left </button> 
        <button type = "button" class="btn1" onclick = "moveSprite(step, 0)" > right </button> 
        <button type = "button" class="btn1" onclick = "moveSprite(0, -step)"> up </button> 
        <button type = "button" class="btn1" onclick = "moveSprite(0, step)" > dn </button> 
          Drehen: 
        <button type = "button" class="btn1" onclick = "rotateSprite(+rwin)"> +45° </button> 
        <button type = "button" class="btn1" onclick = "rotateSprite(-rwin)"> -45° </button> 
          Spiegeln: 
        <button type = "button" class="btn1" onclick = "mirrorSprite(0)" > x </button> 
        <button type = "button" class="btn1" onclick = "mirrorSprite(1)" > y </button> 
        </fieldset>
  </form>
</body>
</html>
```

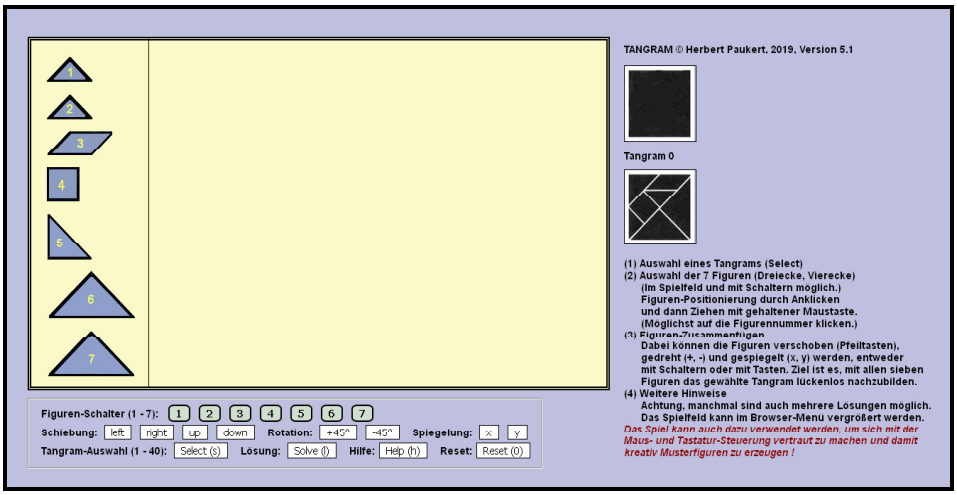
• Sprites running (running.html)

Hinweis: In diesem Programm werden acht Sprites, welche aufeinanderfolgende Phasen einer Bewegung darstellen in kurzen Zeitintervallen dargeboten.

```
<!DOCTYPE html>
<html>
<head>
<title>running.html</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Sprites-Animation">
<meta name="author" content="H.P">

<script>

// ---------- globale Variable (Beginn)

  var max = 7;                    // Anzahl der Sprites (0,1,...,6,7)
  var frame = 0;                  // Sprite-Nummer
  var name0 = "run";              // Sprite-Namensanfang
  var imgList = new Array(max);   // Array der Sprites
  var sprite;                     // Sprite-Objekt
  var spriteImage;                // Image-Objekt
  var MAX_X = 500;                // Spielfeld-Begrenzung
  var zeit = 100;                 // Zeit zwischen den Sprite-Anzeigen
  var step = 10;                  // Distanz zwischen den Sprite-Anzeigen
  var tvar;                       // Hilfsvariable
  var run = false;                // Steuervariable für Start/Stop

// ---------- globale Variable (Ende)

  function init(){
    for (var i = 0; i <= max; i++) {
      name = name0 + i + '.gif';
      imgList[i] = name;
    }
    sprite = document.getElementById("sprite");
    spriteImage = document.getElementById("image");
    document.onmousedown = mDown;
  }

  function animate(){
    updateImage();
    updatePosition();
  }

  function updateImage(){
    frame++;
    if (frame > max) { frame = 0; }
    spriteImage.src = imgList[frame];
  }

  function updatePosition(){
    var x = parseInt(sprite.style.left);
    x += step;
    if (x > MAX_X) { x = 0; }
    sprite.style.left = x + "px";
  }

  function mDown() {
    run = !run;
    if ( run )  { tvar = setInterval("animate()", zeit); }
    if ( !run ) { window.clearInterval(tvar); }
  }

</script>
</head>

<body onload = "init()">
  <h3> Running sprites (start/stop with mouseClick)</h3>
  <div id = "sprite" style = "position: absolute; top: 100px; left: 100px;">
    <img src = "run0.gif" id = "image" alt = ""/>
  </div>
</body>
</html>
```

[6.2] Das Figurenspiel TANGRAM

Hinweis: Das folgende Programm realisiert das bekannte Spiel TANGRAM. Dabei müssen mit Hilfe von sieben vorgegebenen Sprites (Dreiecke und Vierecke) bestimmte grafische Figuren (so genannte Tangrams) erzeugt werden. Im vorliegenden Fall können 40 solche Tangrams als Grafikdateien geladen werden – und auch ihre Lösungsbilder.




```
<!DOCTYPE html">
<html>
<head>
<title>tangram.html </title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=0.90, user-scalable=no">
<meta name="description" content="Tangram-Spielprogramm">
<meta name="author" content="Herbert Paukert">

<style>
body  { background: #BFBFDF; color: black; font-family: sans-serif; font-size: 18px; font-weight: bold;
        -ms-user-select: None; -moz-user-select: None; -webkit-user-select: None; user-select: None; }
.btn2 { border: 2px solid #000000; border-radius: 25%; color: black; background: #D0DFD0;
        font-size: 26px; font-weight: bold; margin: 2px; }
.btn1 { border: 1px solid #000000; color: black; background: #FFFFFF;
        font-size: 26px; margin: 2px; }
#sprite1 {
  position: absolute;
  left:   20px;
  top:    20px;
  width:  70px;
  height: 40px;
  background-image: url('tan1.png');
}
#sprite2 {
  position: absolute;
  left:   20px;
  top:    70px;
  width:  70px;
  height: 40px;
  background-image: url('tan2.png');
}
#sprite3 {
  position: absolute;
  left:   20px;
  top:   120px;
  width: 100px;
  height: 40px;
  background-image: url('tan3.png');
}
#sprite4 {
  position: absolute;
  left:   20px;
  top:   170px;
  width:  50px;
  height: 50px;
  background-image: url('tan4.png');
}
#sprite5 {
  position: absolute;
  left:   20px;
  top:   230px;
  width:  70px;
  height: 70px;
  background-image: url('tan5.png');
}
```

```
#sprite6 {
  position: absolute;
  left:    20px;
  top:    310px;
  width:  130px;
  height:  70px;
  background-image: url('tan6.png');
}
#sprite7 {
  position: absolute;
  left:    20px;
  top:    390px;
  width:  130px;
  height:  70px;
  background-image: url('tan7.png');
}
#surface {
  position: absolute;
  background-color: rgb(250,250,200);
  border: black 5px double;
  left:    10px;
  top:     10px;
  width:  810px;
  height: 470px;
}
#start {
  position: absolute;
  border-right: black 1px solid;
  left:    10px;
  top:     10px;
  width:  170px;
  height: 475px;
}
#controls {
  position: absolute;
  top: 500px;
}
#fs {
  width: 860px;
}

@media screen and (max-width: 640px) {
#picts {
  position:absolute;
  left:10px;
  top:640px;
  width:800px;
}
}
@media screen and (min-width: 641px) {
#picts {
  position:absolute;
  left:840px;
  top:10px;
  width:200px;
}
}
</style>

<script>
   var childWindow;
   var child = false;
   window.addEventListener( "unload", function() { closeChild(); } );

   function closeChild() {
   /* Hilfetext schließen */
     if (!childWindow || childWindow.closed) { return; }
     if (child) { childWindow.document.close(); childWindow.close(); }
   }
   function showHelp() {
   /* Hilfetext anzeigen */
     if (child) { closeChild(); child = false; return; }
     child = true;
     childWindow = window.open('','childWindow','top=25, left=850, width=450, height=680, srollbars=yes, menubar=no,
                               toolbar=no');
     childWindow.document.open();
     childWindow.document.write('<html><head><meta name="viewport" content="width=device-width, initial-
                                 scale=1.0"></head>');
     childWindow.document.write('<body style="background-color:#F8F8F8; font-family: sans-serif; font-size:14px;
                                 text-size-adjust:none; padding-left:10px; padding-bottom:10px;">');
     childWindow.document.write('<br><input type="button" value="Print" onclick="print();"><br><br>');
     childWindow.document.write('Das TANGRAM-Spiel, Version 5.0<br><br>');
     childWindow.document.write('(1) Auswahl eines Tangrams mit Select.<br>');
     childWindow.document.write('(Lösung einblenden mit Solve).<br>');
     childWindow.document.write('(Neustart des Spiels mit Reset).<br><br>');
     childWindow.document.write('(2) Auswahl der 7 Figuren (Dreiecke, Vierecke):<br>');
     childWindow.document.write('Möglichst auf die Figurennummer klicken.<br>');
     childWindow.document.write('Im Spielfeld und mit Schaltern möglich.<br><br>');
     childWindow.document.write('(2a) Figuren-Positionierung mittels Ziehen<br>');
     childWindow.document.write('und Loslassen mit der Maus (Maus-Move).<br>');
     childWindow.document.write('Automatisch bei Desktops mit Widescreens.<br><br>');
     childWindow.document.write('(2b) Figuren-Positionierung mittels Klick-Klick:<br>');
     childWindow.document.write('Erstens durch Anklicken des Figuren-Schalters.<br>');
     childWindow.document.write('Zweitens durch Anklicken der Figuren-Position.<br>');
     childWindow.document.write('(Automatisch bei Phones mit Smallscreens.)<br><br>');
```

```
     childWindow.document.write('(3) Bewegung der Figuren:<br>');
     childWindow.document.write('Dabei können die Figuren verschoben<br>');
     childWindow.document.write('(Pfeiltasten), gedreht (+, -) und auch<br>');
     childWindow.document.write('gespiegelt (x, y) werden, entweder mit<br>');
     childWindow.document.write('Schaltern oder mit Tasten.<br><br>');
     childWindow.document.write('(4) Ziel ist es, mit allen sieben Figuren<br>');
     childWindow.document.write('das gewählte Tangram lückenlos nachzubilden.<br>');
     childWindow.document.write('Dabei sind auch mehrere Lösungen möglich.<br><br>');
     childWindow.document.write('<b>[Klick on/off]</b> wechselt jederzeit zwischen<br>');
     childWindow.document.write('Maus-Move-Technik und Klick-Klick-Technik.<br><br>');
     childWindow.document.write('Hinweis zum Optimieren auf Smartphones:<br>');
     childWindow.document.write('Ev. Verwendung eines Berührungsstiftes!<br><br>');
     childWindow.document.write('(c) Herbert Paukert<br><br>');
     childWindow.document.write('</body></html>');
   }
// --------------------  globale Variable (Beginn) --------------------
var MIN_X =   0;                // Spielfeldgrenzen
var MAX_X = 750;
var MIN_Y =   0;
var MAX_Y = 430;
var MAX_S = 100;                // Koordianten-Ausgleich für das Spielfeld
var dxy = 15;                   // Koordinaten-Ausgleich für die Spielfiguren
var sprite;                     // Sprite-Objekt
var arr = [];                   // Array der Sprites
arr[0] = 0;
var num = 0;                    // Sprite-Nummer
var znum = 0;                   // z-Index des aktuellen Sprites
var anz = 7;                    // Anzahl der Sprite-Objekte
var lSprite = new Array(anz+1);  // Sprite-Left
var tSprite = new Array(anz+1);  // Sprite-Top
var wSprite = new Array(anz+1);  // Sprite-Width
var hSprite = new Array(anz+1);  // Sprite-Height

var smallScreen = 640;          // Screen von Smartphones
var desktop = true;             // Spielvariante mit Mausziehen oder Mausklicken
var run = false;                // Spielvariable

var wied = false;               // Kennvariable für Zugsbeginn (bei Klick-Klick-Technik)
var neu = false;                // Kennvariable für neue Sprite-Figur (bei Klick-Klick-Technik)
var neunum = 0;                 // Neue Sprite-Nummer (bei Klick-Klick-Technik)

var x = 20;                     // x-Koordinate eines Punkes im Spielfeld
var y = 20;                     // y-Koordinate eines Punkes im Spielfeld
var step = 1;                   // Schiebungslänge
var rwin = 45;                  // Rotationswinkel
var rotold = 0;                 // Gespeicherter Rotationswinkel
var mirdir = 0;                 // Gespeicherte Spiegelungsrichtung
var mirold = 1;                 // Wechselschalter für Spiegelungen

var datnum = 1;                 // Muster-Nummer
var datnam = 'tang00.jpg';      // Muster-Name
var datlsg = 'tang_00.jpg';     // Lösungsname
var helpFlag = false;           // Wechselschalter für Hilfstext
// --------------------  globale Variable (Ende) --------------------

// Left, Top, Width und Height der Sprites
for (i = 1; i <= anz; i++) {
   lSprite[i] = 0;
   tSprite[i] = 0;
   wSprite[i] = 0;
   hSprite[i] = 0;
}

function init(){
// Sprites initialisieren
  if (screen.width <= smallScreen) { desktop = false; }
  if (screen.width > smallScreen)  {
     desktop = true;
     var x = document.getElementsByClassName("btn1");
     var y = document.getElementsByClassName("btn2");
     for (i = 0; i < x.length; i++) { x[i].style.fontSize = "18px"; }
     for (i = 0; i < y.length; i++) { y[i].style.fontSize = "18px"; }
     document.getElementById("fs").style.width = "790px";
  }
  for (i = 1; i <= anz; i++) {
    id = 'sprite' + i;
    el = document.getElementById(id);
    arr[i] = el;
    lSprite[i] = parseInt(el.style.left);
    tSprite[i] = parseInt(el.style.top);
    wSprite[i] = parseInt(el.style.width);
    hSprite[i] = parseInt(el.style.height);
//  alert(id + ':' + '\n' + lSprite[i] + ' / ' + tSprite[i] + '\n' + wSprite[i] + ' / ' + hSprite[i]);
  }
  document.onkeydown   = keyListener;
  document.onmousedown = mDown;
  document.onmouseup   = mUp;
  document.onmousemove = mMove;
}

function checkBounds(x,y) {
// Spiefeldrand prüfen für die Mouse-Move-Technik
  if ( (x < MIN_X) || (x > MAX_X) || (y < MIN_Y) || (y > MAX_Y) ) { run = false; return false; }
  else {return true; }
}
```

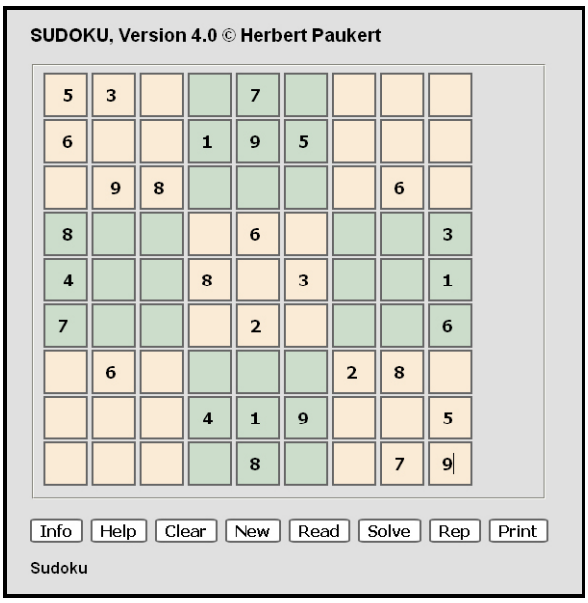
```
function setBounds(x,y) {
// Spiefeld begrenzen für die Klick-Klick-Technik
  if (x < MIN_X){ x = MIN_X; return false; }
  if (x > MAX_X){ x = MAX_X; return false; }
  if (y < MIN_Y){ y = MIN_Y; return false; }
  if (y > MAX_Y){ y = MAX_Y; return false; }
  return true;
}
function getSprite(x,y) {
// Sprite und Sprite-Nummer ermitteln
  var n = 0;
  for (i = 1; i <= anz; i++) {
    a = lSprite[i];
    b = tSprite[i];
    c = wSprite[i];
    d = hSprite[i];
    e = 1*a + 1*c + dxy;
    f = 1*b + 1*d + dxy;
    if ( (x >= a) && (x <= e) && (y >= b) && (y <= f) ) { n = i; break; }
  }
  return n;
}
function checkPlay(x,y) {
// Test, ob Mauscursor im Spielfeld ist
    a = MIN_X;
    b = MIN_Y;
    e = MAX_X + MAX_S;
    f = MAX_Y + MAX_S;
    if ( (x >= a) && (x <= e) && (y >= b) && (y <= f) ) { return true; }
    else { run = false; return false;  }
}
function mDown(event) {
// Sprite-Auswahl mit "mousedown"
if (desktop) {
  run = true;
  x = event.pageX;
  y = event.pageY;
  okay = checkPlay(x,y);
  if (!okay) { run = false; return; }
  num = getSprite(x,y);
  sprite = arr[num];
  znum += 1;
  sprite.style.zIndex = znum;
  return;
}
if (!wied) { // nur aktiv bei Klick-Klick-Technik
  run = true;
  x = event.pageX;
  y = event.pageY;
  innen = checkBounds(x,y);
  if (!innen) { return; }
  okay = checkPlay(x,y);
  if (!okay) { return; }
  num = getSprite(x,y);
  if (neu) { num = neunum; }
  sprite = arr[num];
  znum += 1; sprite.style.zIndex = znum;
  wied = true;
  return;
}
if ( (wied) || (neu) ) { // nur aktiv bei Klick-Klick-Technik
  if (!run) {return; }
  height = parseInt(sprite.style.height);
  width  = parseInt(sprite.style.width);
  x = event.pageX - (width/2);
  y = event.pageY - (height/2);
  okay = setBounds(x,y);
  if (!okay) { return; }
  lSprite[num] = x;
  tSprite[num] = y;
  sprite.style.left = x + "px";
  sprite.style.top = y + "px";
  s = 'scale(1)';
}
}
function mUp(event) {
// Sprite-Bewegung beenden mit "mouseup"
  if (!desktop) { return; }
  run = false;
}
function mMove(event) {
// Sprite-Bewegung ausführen mit "mousemove"
  if (!desktop) { return; }
  if (!run) { return; }
  height = parseInt(sprite.style.height);
  width  = parseInt(sprite.style.width);
  x = event.pageX - (width/2) - dxy
  y = event.pageY - (height/2) - dxy;
  okay = checkBounds(x,y);
  if (!okay) { run = false; return; }
```

```
  lSprite[num] = x;
  tSprite[num] = y;
  sprite.style.left = x + "px";
  sprite.style.top = y + "px";
  s = 'scale(1)';
  if (x < 140) {
     sprite.style.transform = s;
     sprite.style.webkitTransform = s;
  }
}

function keyListener(event){
// Tastatur-Handler
//  alert(event.keyCode);
  if (event.keyCode == 48)  { refresh(); }
  if (event.keyCode == 49)  { num = 1; sprite = arr[num]; }
  if (event.keyCode == 50)  { num = 2; sprite = arr[num]; }
  if (event.keyCode == 51)  { num = 3; sprite = arr[num]; }
  if (event.keyCode == 52)  { num = 4; sprite = arr[num]; }
  if (event.keyCode == 53)  { num = 5; sprite = arr[num]; }
  if (event.keyCode == 54)  { num = 6; sprite = arr[num]; }
  if (event.keyCode == 55)  { num = 7; sprite = arr[num]; }
  if (event.keyCode == 37)  { moveSprite(-step, 0); }    // left
  if (event.keyCode == 38)  { moveSprite(0, -step); }    // up
  if (event.keyCode == 39)  { moveSprite(step, 0);  }    // right
  if (event.keyCode == 40)  { moveSprite(0, step);  }    // down
  if (event.keyCode == 171) { rotateSprite(+rwin); }     // rotation +
  if (event.keyCode == 173) { rotateSprite(-rwin); }     // rotation -
  if (event.keyCode == 88)  { mirrorSprite(0); }         // mirror X
  if (event.keyCode == 89)  { mirrorSprite(1); }         // mirror Y
  if (event.keyCode == 83)  { select(); }                // select
  if (event.keyCode == 76)  { solve();  }                // solve
  if (event.keyCode == 72)  { showHelp(); }              // help
}

function moveSprite(dx, dy){
// Sprite verschieben
  x = parseInt(sprite.style.left);
  y = parseInt(sprite.style.top);
  x += dx;
  y += dy;
  innen = checkBounds(x,y);
  if (!innen) { return; }
  lSprite[num] = x;
  tSprite[num] = y;
  sprite.style.left = x + "px";
  sprite.style.top = y + "px";
  wied = false;
}

function rotateSprite(w) {
// Sprite rotieren (um den Sprite-Mittelpunkt)
  w = rotold + w;
  s = '';
  if ( mirdir == 0) { s = 'rotate(' + w + 'deg)' + ' scaleX(' + mirold + ')' ; }
  if ( mirdir == 1) { s = 'rotate(' + w + 'deg)' + ' scaleY(' + mirold + ')' ; }
  rotold = w;
  sprite.style.transformOrigin = "50% 50%";
  sprite.style.webkitTransformOrigin = "50% 50%";
  sprite.style.transform = s;
  sprite.style.webkitTransform = s;
  wied = false;
}

function mirrorSprite(r) {
// Sprite spiegeln (an X- oder Y-Achse)
  mirdir = r;
  mirror = (-1) * mirold;
  s = '';
  if ( mirdir == 0) { s = ' scaleX(' + mirror + ')' +  ' rotate(' + rotold + 'deg)'; }
  if ( mirdir == 1) { s = ' scaleY(' + mirror + ')' +  ' rotate(' + rotold + 'deg)'; }
  mirold = mirror;
  sprite.style.transformOrigin = "50% 50%";
  sprite.style.webkitTransformOrigin = "50% 50%";
  sprite.style.transform = s;
  sprite.style.webkitTransform = s;
  wied = false;
}

function clickSprite(node) {
// Zusätzliche Sprite-Auswahl mittels Zahlen-Schalter (Klick-Klick-Technik)
  s = node.id;
  t = s.substr(2,1)
  num = parseInt(t);
  sprite = arr[num];
  neunum = num;
  neu = true;
  wied = false;
}

function refresh() {
// Reset
  window.location.reload();
}
```

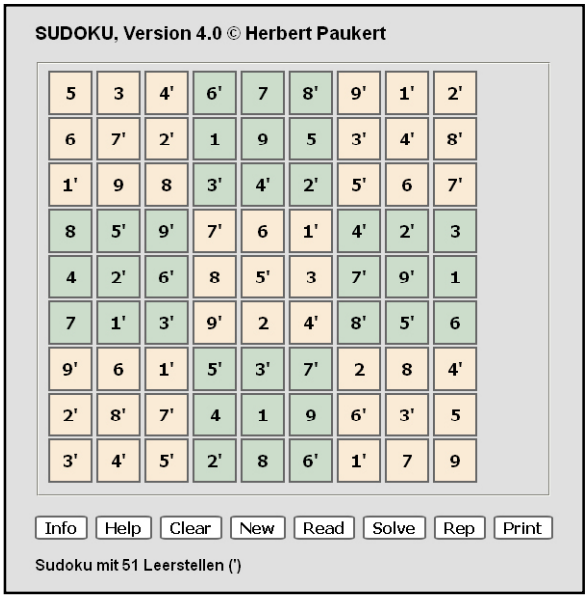
```
function select() {
// Tangram-Auswahl
  var pict = document.getElementById("pix0");
  var loes = document.getElementById("pix1");
  s = prompt('Tangram-Nummer (1 - 40)',datnum);
  if ( isNaN(s) ) { s = 1; }
  if ((s < 1) || (s > 40)) { datnum = 1; }
  else { datnum = s; }
  if (datnum < 10) {
     datnam = 'tang0' + datnum + '.jpg';
     datlsg = 'tang_0' + datnum + '.jpg';
  }
  else {
     datnam = 'tang' + datnum + '.jpg';
     datlsg = 'tang_' + datnum + '.jpg';
  }
  pict.src = datnam;
  loes.src = ' ';
  document.getElementById("nr").innerHTML = 'Tangram ' + datnum;
}

function solve() {
// Tangram-Lösung
  var pict = document.getElementById("pix0");
  var loes = document.getElementById("pix1");
  ok = confirm('Lösung wirklich anzeigen ?');
  if ( !ok ) { return; }
  loes.src = datlsg;
}

function clickDesk() {
// Klick-Move-Toggle (Wechsel zwischen Mouse_Move und Klick-Klick)
   desktop = !desktop;
   if (desktop)  { alert('Positionieren mit Maus-Move !'); }
   if (!desktop) { alert('Positionieren mit Klick-Klick !'); }
}

</script>
</head>

<body onload = "init()">
  <div id = "surface">
    <div id="sprite1" style="position:absolute; left:20px; top:20px; width:70px; height:40px;"></div>
    <div id="sprite2" style="position:absolute; left:20px; top:70px; width:70px; height:40px;"></div>
    <div id="sprite3" style="position:absolute; left:20px; top:120px; width:100px; height:40px;"></div>
    <div id="sprite4" style="position:absolute; left:20px; top:170px; width:50px; height:50px;"></div>
    <div id="sprite5" style="position:absolute; left:20px; top:230px; width:70px; height:70px;"></div>
    <div id="sprite6" style="position:absolute; left:20px; top:310px; width:130px; height:70px;"></div>
    <div id="sprite7" style="position:absolute; left:20px; top:390px; width:130px; height:70px;"></div>
  </div>

  <div id = "start">
  </div>

  <form action = "" id = "controls">
      <fieldset id = "fs">
          Figuren-Schalter (1 - 7):  
        <button type = "button" class="btn2" id="b71" onclick="clickSprite(this)">1</button> 
        <button type = "button" class="btn2" id="b72" onclick="clickSprite(this)">2</button> 
        <button type = "button" class="btn2" id="b73" onclick="clickSprite(this)">3</button> 
        <button type = "button" class="btn2" id="b74" onclick="clickSprite(this)">4</button> 
        <button type = "button" class="btn2" id="b75" onclick="clickSprite(this)">5</button> 
        <button type = "button" class="btn2" id="b76" onclick="clickSprite(this)">6</button> 
        <button type = "button" class="btn2" id="b77" onclick="clickSprite(this)">7</button> 
          Neustart: <button type = "button" class="btn1" onclick="refresh()"> Reset </button> 
        <br>
          Schieben: 
        <button type = "button" class="btn1" onclick = "moveSprite(-step, 0)"> left  </button> 
        <button type = "button" class="btn1" onclick = "moveSprite(step, 0)" > right </button> 
        <button type = "button" class="btn1" onclick = "moveSprite(0, -step)"> up </button> 
        <button type = "button" class="btn1" onclick = "moveSprite(0, step)" > dn </button> 
          Drehen: 
        <button type = "button" class="btn1" onclick = "rotateSprite(+rwin)"> +45° </button> 
        <button type = "button" class="btn1" onclick = "rotateSprite(-rwin)"> -45° </button> 
          Spiegeln: 
        <button type = "button" class="btn1" onclick = "mirrorSprite(0)" > x </button> 
        <button type = "button" class="btn1" onclick = "mirrorSprite(1)" > y </button> 
        <br>
          Tangram (1 - 40): 
        <button type = "button" class="btn1" onclick="select()"> Select </button> 
          Lösung: 
        <button type = "button" class="btn1" onclick="solve()"> Solve </button> 
          Klick: <button type = "button" class="btn1" onclick="clickDesk()"> on/off </button> 
          Hilfe: <button type = "button" class="btn1" onclick="showHelp()"> on/off </button> 
      </fieldset>
  </form>

  <div id = "picts">
    <img id = "pix0" src = "tang00.jpg" style="margin:10px; height:150px; float: left;" alt="Tangram">
    <img id = "pix1" src = "tang_00.jpg" style="margin:10px; height:150px; float:clear;" alt= " ">
    <p id = "nr" style="margin:10px;">Tangram 0</p>
  </div>

</body>
</html>
```

[6.3] Das Zahlenspiel SUDOKU

Hinweis: Dieses Programm beschäftigt sich mit dem japanischen Zahlenspiel SUDOKU.
Zuerst folgt eine allgemeine Spielbeschreibung und dann das Programmlisting.



Eingegebenes Sudoku



Gelöstes Sudoku

Ein SUDOKU (japanisches Zahlenrätsel) besteht aus einem Raster mit 9 x 9 = 81 Feldern. Zusätzlich ist dieser Raster in neun quadratische Blöcke unterteilt mit 3 x 3 = 9 Feldern. In jedem Feld sitzt eine Ziffer 1,2,...,9. Einige Felder sind unbesetzt. Die Aufgabe eines Spielers ist es, in diese unbesetzten Felder die Ziffern so einzutragen, dass folgende Spielregeln gelten:

(1) In der waagrechten Zeile durch das Feld,
(2) in der senkrechten Spalte durch das Feld,
(3) in dem quadratischen Block um das Feld
darf jede Ziffer nur genau EINMAL vorkommen!

Das vorliegende Programm bietet mehrere Möglichkeiten:
[*Clear*] manuelle Eingabe eines neuen Rätsels
[*New*] automatische Erzeugung von neuen Rätseln
[*Read*] Auswahl von drei vorgegebenen Rätseln
[*Solve*] automatische Auflösung eines Rätsels
[*Rep*] Wiederholung von jedem neuen Rätsel

MANUELLE LÖSUNGSMETHODEN

Die Methoden sollen am dargestellten SUDOKU demonstriert werden.

```
|5 3 .|. 7 .|. . .|
|6 . .|1 9 5|. . .|
|. 9 8|. . .|. 6 .|
|-----+-----+-----|
|8 . .|. 6 .|. . 3|
|4 . .|8 . 3|. . 1|
|7 . .|. 2 .|. . 6|
|-----+-----+-----|
|. 6 .|. . .|2 8 .|
|. . .|4 1 9|. . 5|
|. . .|. 8 .|. 7 9|
```

```
|5 3 4|6 7 8|9 1 2|
|6 7 2|1 9 5|3 4 8|
|1 9 8|3 4 2|5 6 7|
|-----+-----+-----|
|8 5 9|7 6 1|4 2 3|
|4 2 6|8 5 3|7 9 1|
|7 1 3|9 2 4|8 5 6|
|-----+-----+-----|
|9 6 1|5 3 7|2 8 4|
|2 8 7|4 1 9|6 3 5|
|3 4 5|2 8 6|1 7 9|
```

Sudokus mit vielen Leerstellen sind immer schwieriger aufzulösen.

[1] Die Ausschluss-Methode

Suchen Sie eine Zeile oder eine Spalte oder einen Block mit den wenigsten Leerfeldern. Bestimmen Sie dort alle fehlenden Ziffern und schreiben Sie diese dort in jedes leere Feld. (z.B. stehen dann in allen Leerfeldern der fünften Spalte die Ziffern {3,4,5}. Nun können Sie in jedem dieser Felder auf Grund der restlichen Informationen in den zugeordneten Zeilen und Blöcken bestimmte Ziffern streichen. (So stehen dann z.B. in der 3. Zeile nur mehr {3,4}, in der 5. Zeile nur {5} und in der 7. Zeile nur {3,5}). Wenn Sie nun diese Felder noch einmal durchlaufen, so schließen sich wieder bestimmte Ziffern aus, bis schließlich nur mehr eine Ziffer in jedem Feld steht. (z.B. 4 in der dritten, 5 in der fünften und 3 in der siebenten Zeile von der fünften Spalte). Damit ist diese Spalte komplett und die erhaltenen richtigen Ziffern werden eingeringelt. Dann wendet man sich der nächsten Zeile, Spalte oder Block mit möglichst wenigen Leerfeldern zu. Je nach Ziffernanordnung muss man zunächst in manchen Feldern zwei oder mehrere Ziffern stehen lassen, ehe klar wird, welche Ziffer die richtige ist.

[2] Die Kombinations-Methode

Enthalten zwei Blöcke einer Reihe die gleiche Ziffer, dann kann die Position dieser Ziffer im dritten Block dieser Reihe oft einfach ermittelt werden. (z.B. die Ziffer 5 im linken oberen und mittleren oberen Block. Diese Ziffer kann im rechten oberen Block nur in der 3. Zeile stehen. Dort kann sie nur in der 7. Spalte stehen, weil sie ja in der 9. Spalte vorkommt. Damit befindet sich die Ziffer 5 in der 3. Zeile und 7. Spalte).

[3] Die Kandidaten-Methode

Suchen Sie eine Zeile oder eine Spalte oder einen Block mit den wenigsten Leerfeldern. Betrachten Sie dort ein bestimmtes Feld. Ermitteln Sie für dieses Feld alle erlaubten Ziffern, d.h alle Ziffern, welche nicht in zugeordneten Block, Spalte und Zeile vorkommen. Schreiben Sie diese Kandidaten-Menge in das Feld und ermitteln Sie die Kandidaten-Menge für ein anderes Feld. Wählen Sie die Felder dabei so aus, dass die jeweilgen Kandidaten-Mengen möglichst klein sind. Wenn dann nur mehr ein einziger Kandidat übrig bleibt, dann haben Sie die richtige Ziffer für das Feld gefunden.

Es sei M die Menge aller Ziffern, also M = {1,2,3,4,5,6,7,8,9}. Betrachten wir nun im Beispiel das Feld in der 7. Zeile und der 9. Spalte und bezeichnen es kurz mit [7/9]. Die Ziffern in der 7. Zeile sind Z(7) = {2,6,8}, die Ziffern in der 9. Spalte sind S(9) = {1,3,5,6,9} und die Ziffern im rechten unteren Block sind B(3,3) = {2,5,7,8,9}. Um die möglichen Ziffern für unser Feld zu erhalten, bilden wir zuerst die entsprechenden Differenzmengen:

```
M \ Z(7)   = {1,3,4,5,7,9}
M \ S(9)   = {2,4,7,8}
M \ B(3,3) = {1,3,4,6}
```

Zuletzt bilden wir von den Mengen die Durchschnittsmenge D = {4} und erhalten das Ergebnis, dass im Feld [7/9] die Ziffer 4 steht.

Erhält man mit dieser Methode mehrelementige Kandidaten-Mengen, dann muss aus der kleinsten Menge ein Kandidat ausgewählt und seine Richtigkeit, durch Vergleich mit den Kandidaten-Mengen in anderen Feldern, schrittweise überprüft werden.

```
<!doctype html>
<html>
<head>
  <title> sudoku.html </title>
  <meta charset="ISO-8859-1">
  <meta name="description" content="Ein japanisches Zahlenrätsel">
  <meta name="author" content="Herbert Paukert">

  <style>
    body  { background-color:#E0E0E0; color:black;
            font-family:Arial; font-size:16px; font-weight:bold; text-size-adjust: none;
            text-align:left; margin:30px; }
    #fs   { width: 510px; }
    .ein1 { border:2px solid #666666; background-color:antiquewhite; font-size:18px;
            font-weight:bold;
            width: 40px; height:40px; margin-top:2px; text-align: center; }
    .ein2 { border:2px solid #666666; background-color:#CCDDCC; font-size:18px; font-weight:bold;
            width: 40px; height:40px; margin-top:2px; text-align: center; }
    .btn  { border:2px solid #666666; background-color:white; font-size:16px; font-weight:bold;
            border-radius:10%; }
    #info { position: absolute; left: 620px; top:70px; background-color:white;font-size:14px;
            padding: 20px; border:1px solid #666666; visibility: hidden; }
  </style>

  <script>
  /* globale Variable, Beginn */

  var arr = new Array();

  var sudo01 = new Array();      // mit 40 Leerstellen
      sudo01[0] = '094230100';
      sudo01[1] = '075906080';
      sudo01[2] = '100050009';
      sudo01[3] = '210009058';
      sudo01[4] = '806020907';
      sudo01[5] = '740600031';
      sudo01[6] = '500090004';
      sudo01[7] = '030102596';
      sudo01[8] = '007064800';

  var sudo02 = new Array();       // mit 51 Leerstellen
      sudo02[0] = '530070000';
      sudo02[1] = '600195000';
      sudo02[2] = '098000060';
      sudo02[3] = '800060003';
      sudo02[4] = '400803001';
      sudo02[5] = '700020006';
      sudo02[6] = '060000280';
      sudo02[7] = '000419005';
      sudo02[8] = '000080079';

  var sudo03 = new Array();       // mit 55 Leerstellen
      sudo03[0] = '910000050';
      sudo03[1] = '080700002';
      sudo03[2] = '060001000';
      sudo03[3] = '050030061';
      sudo03[4] = '000409000';
      sudo03[5] = '870060090';
      sudo03[6] = '000800070';
      sudo03[7] = '300002040';
      sudo03[8] = '040000016';

  var max = 8;
  var mat = new Array();     // Spielfeldraster (Matrix)
  var cop = new Array();     // Matrix-Kopie

  for (i = 0; i <= max; i++) { mat[i] = new Array(max); }
  for (i = 0; i <= max; i++) { cop[i] = new Array(max); }

  for (i = 0; i <= max; i++) {
      for (j = 0; j <= max; j++) {
          mat[i][j] = 0;
          cop[i][j] = 0;
      }
  }

  var count = 0;             // Anzahl der Leerstellen
  var helpFlag = false;      // Wechselschalter für Hilfstext
  var okay = false;          // Steuervariable für "solve"

  /* globale Variable, Ende */
```

```
function clr()
// Spielfeld löschen
{
   document.getElementById('num').innerHTML = 'Sudoku ';
   for (i = 0; i <= max; i++) {
     for (j = 0; j <= max; j++) {
        x = 'i' + 1*(i+1) + 1*(j+1);
        document.getElementById(x).value = ' ';
        mat[i][j] = 0;
     }
   }
   okay = true;
}

function createMatrix()
// das Spielfeld in eine 9x9-Matrix transformieren
{
   count = 0;
   for (i = 0; i <= max; i++) {
     for (j = 0; j <= max; j++) {
        x = 'i' + 1*(i+1) + 1*(j+1);
        y = document.getElementById(x).value;
        z = parseInt(y);
        if ( isNaN(z) ) { z = 0; count++; }
        mat[i][j] = z;
        cop[i][j] = z;
     }
   }
   document.getElementById('num').innerHTML = 'Sudoku mit ' + count + ' Leerstellen';
}

function showMatrix()
// die 9x9-Matrix in das Spielfeld transformieren vor "solve"
{
   count = 0;
   for (i = 0; i <= max; i++) {
     for (j = 0; j <= max; j++) {
        z = mat[i][j];
        y = z;
        if (z == 0) { y = ' '; count++; }
        x = 'i' + 1*(i+1) + 1*(j+1);
        document.getElementById(x).value = y;
     }
   }
   document.getElementById('num').innerHTML = 'Sudoku mit ' + count + ' Leerstellen';
}

function showMatrix1()
// die 9x9-Matrix in das Spielfeld transformieren nach "solve"
{
   count = 0;
   for (i = 0; i <= max; i++) {
     for (j = 0; j <= max; j++) {
        z = mat[i][j];
        y = z;
        z1 = cop[i][j];
        if (z1 == 0) { y = z + "'"; count++; }
        x = 'i' + 1*(i+1) + 1*(j+1);
        document.getElementById(x).value = y;
     }
   }
   document.getElementById('num').innerHTML = "Sudoku mit " + count + " Leerstellen (')";
}

function getGame()
// Drei vorgegebene Spiele auswählen
{
   document.getElementById('num').innerHTML = 'Sudoku ';
   wahl = 1;
   wahl = prompt('Wählen Sie ein Sudoku aus (1 bis 3)',wahl);
   if ( isNaN(wahl) ) { wahl = 1; }
   if ( (wahl < 1) || (wahl > 3) ) { wahl = 1; }
   setGame(wahl);
}
```

```
function setGame(w)
// die ausgewählten Spiele anzeigen und in die Matrix transformieren
{
   if (w == 1) { for (i = 0; i <= max; i++) { arr[i] = sudo01[i]; } }
   if (w == 2) { for (i = 0; i <= max; i++) { arr[i] = sudo02[i]; } }
   if (w == 3) { for (i = 0; i <= max; i++) { arr[i] = sudo03[i]; } }
   count = 0;
   for (i = 0; i <= max; i++) {
     s = arr[i];
     for (j = 0; j <= max; j++) {
        c = s.charAt(j);
        d = c;
        if (c == '0') { d = ' '; count++; }
        x = 'i' + 1*(i+1) + 1*(j+1);
        document.getElementById(x).value = d;
        mat[i][j] = 1*c;
     }
   }
   document.getElementById('num').innerHTML = ' Sudoku ' + w + ' mit ' + count +
                                              ' Leerstellen';
   okay = true;
}

function goSolve()
// Sudoku lösen
{
  if ( !okay ) { alert('Zuerst eine neues Spiel eingeben!'); return; }
  alert(' Weiter zur Lösung . . . . . ');
  createMatrix();
  erg = solve(mat);
  if ( !erg ) {
     alert('Sudoku nicht gelöst !');
  }
  else {
     alert('Sudoku gelöst !');
     showMatrix1();
  }
  okay = false;
}

function solve(mat) {
// Sudoku-Auflösungsfunktion
   for (var i = 0; i <= max; i++) {
     for (var j = 0; j <= max; j++) {
       if (mat[i][j] != 0) { continue; }
       for (var k = 1; k <= 9; k++) {
         if (insert(mat,i,j,k) == true) {
            mat[i][j] = k;
            b = solve(mat);  // rekursiver Funktionsaufruf
            if (b == true) { return true; }
            mat[i][j] = 0;
         }
       }
       return false;
     }
   }
   return true;
 }

 function insert(mat,i,j,k)
 // die Zeilen und Spalten prüfen
 {
   for (var a = 0; a <= max; a++) {
     if (a != i && mat[a][j] == k) { return false; }
   }
   for (var a = 0; a <= max; a++) {
     if (a != j && mat[i][a] == k) { return false; }
   }
   // die 3x3-Untertmatrizen prüfen
   var y = Math.floor((i / 3)) * 3;
   var x = Math.floor((j / 3)) * 3;
   for (var a = 0; a < 3; a++) {
     for (var b = 0; b < 3; b++) {
       if (a != i && b != j && mat[y + a][x + b] == k) { return false; }
     }
   }
   return true;
 }
```

```
function createGame()
// Zufalls-Sudokus erzeugen
{
  document.getElementById('num').innerHTML = 'Sudoku ';
  clr();
  solve(mat);

  // Zeilen und Spalten tauschen (6-mal)

  var a,b,c;
  var z = Math.round(3*Math.random());
  if (z == 1) {
     for (j = 0; j <= max; j++) {
         a = mat[0][j]; mat[0][j] = mat[1][j]; mat[1][j] = a;
         b = mat[4][j]; mat[4][j] = mat[5][j]; mat[5][j] = b;
         c = mat[6][j]; mat[6][j] = mat[8][j]; mat[8][j] = c;
     }
     for (j = 0; j <= max; j++) {
         a = mat[j][0]; mat[j][0] = mat[j][1]; mat[j][1] = a;
         b = mat[j][4]; mat[j][4] = mat[j][5]; mat[j][5] = b;
         c = mat[j][6]; mat[j][6] = mat[j][8]; mat[j][8] = c;
     }
  }

  if (z == 2) {
     for (j = 0; j <= max; j++) {
         a = mat[0][j]; mat[0][j] = mat[2][j]; mat[2][j] = a;
         b = mat[3][j]; mat[3][j] = mat[5][j]; mat[5][j] = b;
         c = mat[7][j]; mat[7][j] = mat[8][j]; mat[8][j] = c;
     }
     for (j = 0; j <= max; j++) {
         a = mat[j][0]; mat[j][0] = mat[j][2]; mat[j][2] = a;
         b = mat[j][3]; mat[j][3] = mat[j][5]; mat[j][5] = b;
         c = mat[j][7]; mat[j][7] = mat[j][8]; mat[j][8] = c;
     }
  }

  if (z == 3) {
     for (j = 0; j <= max; j++) {
         a = mat[1][j]; mat[1][j] = mat[2][j]; mat[2][j] = a;
         b = mat[3][j]; mat[3][j] = mat[4][j]; mat[4][j] = b;
         c = mat[6][j]; mat[6][j] = mat[7][j]; mat[7][j] = c;
     }
     for (j = 0; j <= max; j++) {
         a = mat[j][1]; mat[j][1] = mat[j][2]; mat[j][2] = a;
         b = mat[j][3]; mat[j][3] = mat[j][4]; mat[j][4] = b;
         c = mat[j][6]; mat[j][6] = mat[j][7]; mat[j][7] = c;
     }
  }

  // Blanks einsetzen (9-mal)

  // linke Blocks
   for (i = 0; i <= 2; i++) {
      for (j = 0; j <= 2; j++) {
          z = Math.round(1*Math.random());
          if (z == 0) { mat[i][j] = 0; }
      }
  }
  for (i = 3; i <= 5; i++) {
      for (j = 0; j <= 2; j++) {
          z = Math.round(1*Math.random());
          if (z == 0) { mat[i][j] = 0; }
      }
  }
  for (i = 6; i <= 8; i++) {
      for (j = 0; j <= 2; j++) {
          z = Math.round(1*Math.random());
          if (z == 0) { mat[i][j] = 0; }
      }
  }

  // mittlere Blocks
  for (i = 0; i <= 2; i++) {
      for (j = 3; j <= 5; j++) {
          z = Math.round(1*Math.random());
          if (z == 0) { mat[i][j] = 0; }
      }
  }
```

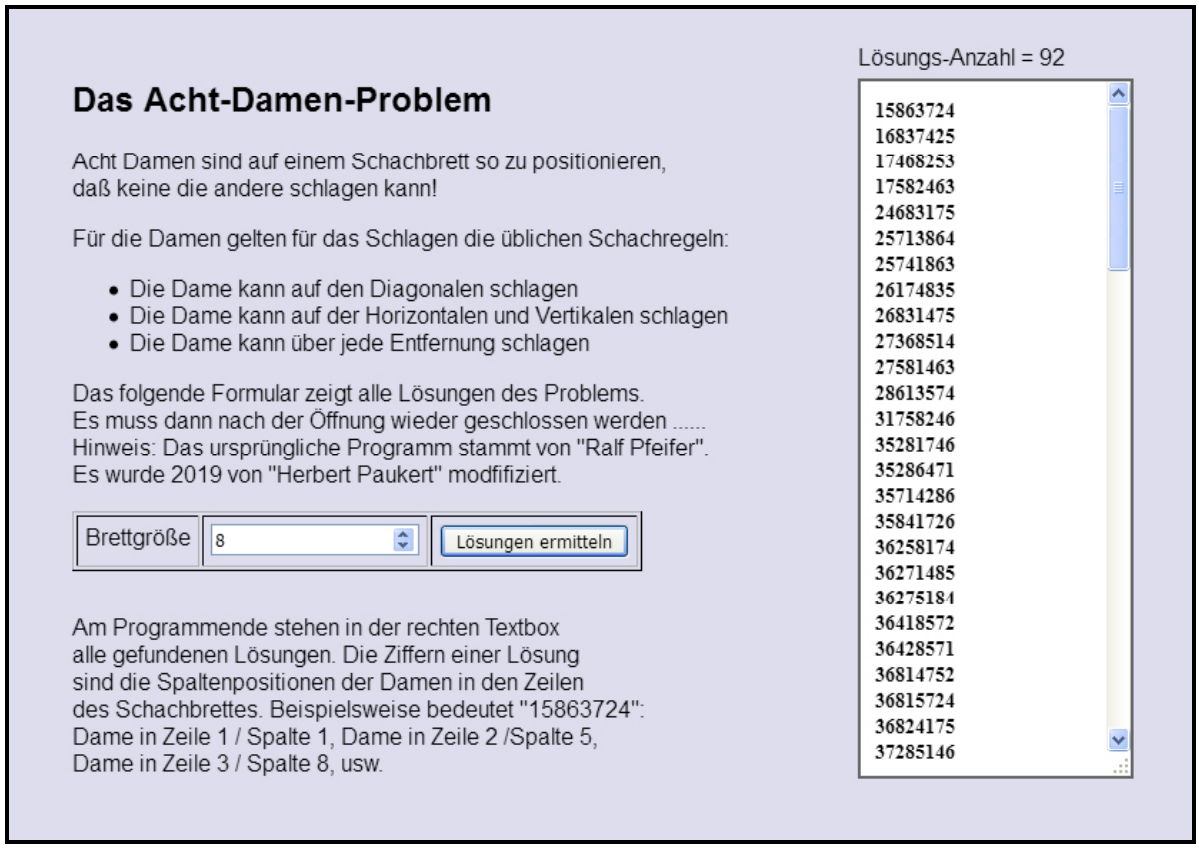
```
      for (i = 3; i <= 5; i++) {
          for (j = 3; j <= 5; j++) {
              z = Math.round(1*Math.random());
              if (z == 0) { mat[i][j] = 0; }
          }
      }
      for (i = 6; i <= 8; i++) {
          for (j = 3; j <= 5; j++) {
              z = Math.round(1*Math.random());
              if (z == 0) { mat[i][j] = 0; }
          }
      }

      // rechte Blocks
      for (i = 0; i <= 2; i++) {
          for (j = 6; j <= 8; j++) {
              z = Math.round(1*Math.random());
              if (z == 0) { mat[i][j] = 0; }
          }
      }
      for (i = 3; i <= 5; i++) {
          for (j = 6; j <= 8; j++) {
              z = Math.round(1*Math.random());
              if (z == 0) { mat[i][j] = 0; }
          }
      }
      for (i = 6; i <= 8; i++) {
          for (j = 6; j <= 8; j++) {
              z = Math.round(1*Math.random());
              if (z == 0) { mat[i][j] = 0;  }
          }
      }

      for (i = 0; i <= max; i++) {
          for (j = 0; j <= max; j++) {
              cop[i][j] = mat[i][j];
          }
      }

      showMatrix();
      okay = true;
    }


    function repeatGame()
    // Neues Spiel wiederholen
    {
      count = 0;
      for (i = 0; i <= max; i++) {
        for (j = 0; j <= max; j++) {
          mat[i][j] = cop[i][j];
          if ( mat[i][j] == 0 ) { count++; }
         }
      }

      showMatrix();
      okay = true;
    }


    function help()
    // Hilfstext ein- ausblenden
    {
     helpFlag = !helpFlag;
     if (helpFlag) { document.getElementById("info").style.visibility = 'visible'; }
     else { document.getElementById("info").style.visibility = 'hidden'; }
    }


    function info()
    // Zu einer zusätzlichen Informationsseite wechseln
    {
      window.location = "sudinfo.html";
    }

    </script>

</head>
```

```
<body>
<h3>SUDOKU, Version 4.0 &copy Herbert Paukert</h3>
<div id = "info">
  Ein SUDOKU (Zahlenrätsel) besteht aus einem Raster mit<br>
  9 x 9 = 81 Feldern. Zusätzlich ist der Raster in genau neun<br>
  quadratische Blöcke unterteilt mit jeweils 3 x 3 = 9 Feldern.<br>
  In jedem Feld sitzt eine Ziffer 1, 2, ..., 9. Einige Felder sind<br>
  noch unbesetzt. Die Aufgabe eines Spielers ist es nun, in<br>
  die unbesetzten Felder die Ziffern 1, 2, ..., 9 so einzutragen,<br>
  dass folgende drei Spielregeln gelten:<br>
  <br><i>
  (1) In der waagrechten Zeile durch das Feld,<br>
  (2) in der senkrechten Spalte durch das Feld,<br>
  (3) in dem quadratischen Block um das Feld<br>
  <font color = "red">
  darf jede Ziffer nur genau EINMAL vorkommen!<br>
  <br></font></i>
  Sudokus mit vielen Leerfeldern sind schwieriger zu lösen.<br>
  Das vorliegende Programm bietet mehrere Möglichkeiten:<br>
  <br>
  [<i><u> Clear </u></i>]  manuelle Eingabe eines neuen Rätsels<br>
  [<i><u> New </u></i>]    automatische Erzeugung von neuen Rätseln<br>
  [<i><u> Read </u></i>]   Auswahl von drei vorgegebenen Rätseln<br>
  [<i><u> Solve </u></i>]  automatische Auflösung eines Rätsels<br>
  [<i><u> Rep </u></i>]     Wiederholung von jedem neuen Rätsel<br>
  <br>
  Hinweis: Bei selbst erstellten Sudokus mit Eingabefehlern<br>
  wie falsche Ziffern oder falsche Ziffernwiederholungen ist<br>
  keine richtige Auflösung möglich!<br>
  <br>
  Hinweis: Zum manuellen Auflösen eines Sudokus kann es<br>
  gedruckt und dann auf dem Papierblatt gelöst werden.<br>
  Die verschiedenen Lösungsverfahren können mit [<i><u> Info </u></i>]<br>
  auf der Internetseite "sudinfo.html" nachgelesen werden.<br>
</div>

<form name="formu">
<fieldset id="fs">
<input type="text" id="i11" class="ein1" value=" " size="1">
<input type="text" id="i12" class="ein1" value=" " size="1">
<input type="text" id="i13" class="ein1" value=" " size="1">
<input type="text" id="i14" class="ein2" value=" " size="1">
<input type="text" id="i15" class="ein2" value=" " size="1">
<input type="text" id="i16" class="ein2" value=" " size="1">
<input type="text" id="i17" class="ein1" value=" " size="1">
<input type="text" id="i18" class="ein1" value=" " size="1">
<input type="text" id="i19" class="ein1" value=" " size="1">
<br>
<input type="text" id="i21" class="ein1" value=" " size="1">
<input type="text" id="i22" class="ein1" value=" " size="1">
<input type="text" id="i23" class="ein1" value=" " size="1">
<input type="text" id="i24" class="ein2" value=" " size="1">
<input type="text" id="i25" class="ein2" value=" " size="1">
<input type="text" id="i26" class="ein2" value=" " size="1">
<input type="text" id="i27" class="ein1" value=" " size="1">
<input type="text" id="i28" class="ein1" value=" " size="1">
<input type="text" id="i29" class="ein1" value=" " size="1">
<br>
<input type="text" id="i31" class="ein1" value=" " size="1">
<input type="text" id="i32" class="ein1" value=" " size="1">
<input type="text" id="i33" class="ein1" value=" " size="1">
<input type="text" id="i34" class="ein2" value=" " size="1">
<input type="text" id="i35" class="ein2" value=" " size="1">
<input type="text" id="i36" class="ein2" value=" " size="1">
<input type="text" id="i37" class="ein1" value=" " size="1">
<input type="text" id="i38" class="ein1" value=" " size="1">
<input type="text" id="i39" class="ein1" value=" " size="1">
<br>
<input type="text" id="i41" class="ein2" value=" " size="1">
<input type="text" id="i42" class="ein2" value=" " size="1">
<input type="text" id="i43" class="ein2" value=" " size="1">
<input type="text" id="i44" class="ein1" value=" " size="1">
<input type="text" id="i45" class="ein1" value=" " size="1">
<input type="text" id="i46" class="ein1" value=" " size="1">
<input type="text" id="i47" class="ein2" value=" " size="1">
<input type="text" id="i48" class="ein2" value=" " size="1">
<input type="text" id="i49" class="ein2" value=" " size="1">
<br>
```

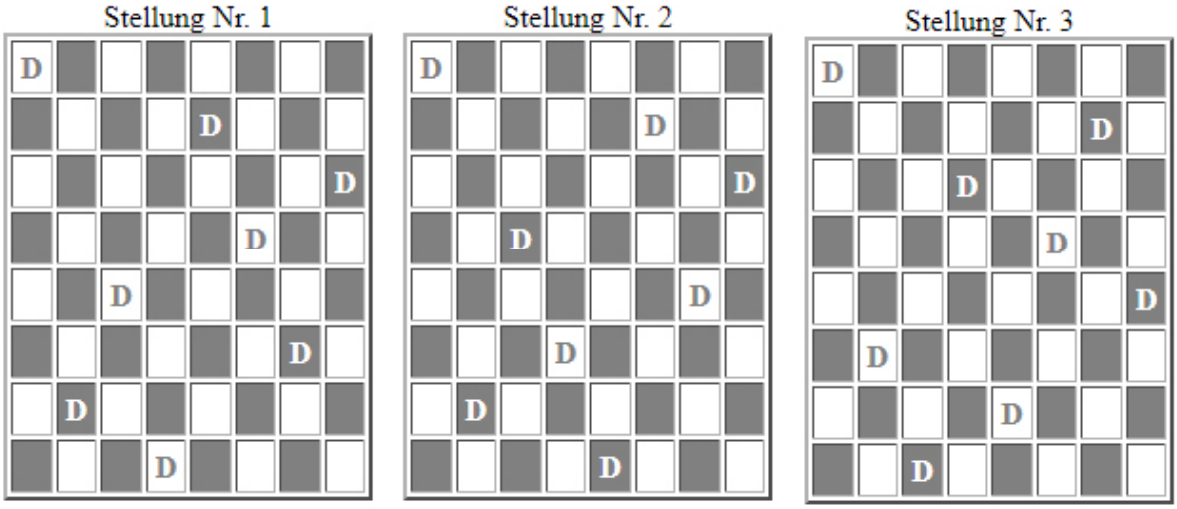
```
<input type="text" id="i51" class="ein2" value=" " size="1">
<input type="text" id="i52" class="ein2" value=" " size="1">
<input type="text" id="i53" class="ein2" value=" " size="1">
<input type="text" id="i54" class="ein1" value=" " size="1">
<input type="text" id="i55" class="ein1" value=" " size="1">
<input type="text" id="i56" class="ein1" value=" " size="1">
<input type="text" id="i57" class="ein2" value=" " size="1">
<input type="text" id="i58" class="ein2" value=" " size="1">
<input type="text" id="i59" class="ein2" value=" " size="1">
<br>
<input type="text" id="i61" class="ein2" value=" " size="1">
<input type="text" id="i62" class="ein2" value=" " size="1">
<input type="text" id="i63" class="ein2" value=" " size="1">
<input type="text" id="i64" class="ein1" value=" " size="1">
<input type="text" id="i65" class="ein1" value=" " size="1">
<input type="text" id="i66" class="ein1" value=" " size="1">
<input type="text" id="i67" class="ein2" value=" " size="1">
<input type="text" id="i68" class="ein2" value=" " size="1">
<input type="text" id="i69" class="ein2" value=" " size="1">
<br>
<input type="text" id="i71" class="ein1" value=" " size="1">
<input type="text" id="i72" class="ein1" value=" " size="1">
<input type="text" id="i73" class="ein1" value=" " size="1">
<input type="text" id="i74" class="ein2" value=" " size="1">
<input type="text" id="i75" class="ein2" value=" " size="1">
<input type="text" id="i76" class="ein2" value=" " size="1">
<input type="text" id="i77" class="ein1" value=" " size="1">
<input type="text" id="i78" class="ein1" value=" " size="1">
<input type="text" id="i79" class="ein1" value=" " size="1">
<br>
<input type="text" id="i81" class="ein1" value=" " size="1">
<input type="text" id="i82" class="ein1" value=" " size="1">
<input type="text" id="i83" class="ein1" value=" " size="1">
<input type="text" id="i84" class="ein2" value=" " size="1">
<input type="text" id="i85" class="ein2" value=" " size="1">
<input type="text" id="i86" class="ein2" value=" " size="1">
<input type="text" id="i87" class="ein1" value=" " size="1">
<input type="text" id="i88" class="ein1" value=" " size="1">
<input type="text" id="i89" class="ein1" value=" " size="1">
<br>
<input type="text" id="i91" class="ein1" value=" " size="1">
<input type="text" id="i92" class="ein1" value=" " size="1">
<input type="text" id="i93" class="ein1" value=" " size="1">
<input type="text" id="i94" class="ein2" value=" " size="1">
<input type="text" id="i95" class="ein2" value=" " size="1">
<input type="text" id="i96" class="ein2" value=" " size="1">
<input type="text" id="i97" class="ein1" value=" " size="1">
<input type="text" id="i98" class="ein1" value=" " size="1">
<input type="text" id="i99" class="ein1" value=" " size="1">
</fieldset>
<br>
</form>
<button id="btn0" class="btn" onclick="info()"> Info </button> 
<button id="btn1" class="btn" onclick="help()"> Help </button> 
<button id="btn2" class="btn" onclick="clr()"> Clear </button> 
<button id="btn3" class="btn" onclick="createGame()"> New </button> 
<button id="btn4" class="btn" onclick="getGame()"> Read </button> 
<button id="btn5" class="btn" onclick="goSolve()"> Solve </button> 
<button id="btn6" class="btn" onclick="repeatGame()"> Rep </button> 
<button id="btn7" class="btn" onclick="print()"> Print </button> 
<p id = "num"> Sudoku </p>
</body>
</html>
```

[6.4] Das Acht-Damen-Problem

Dieses letzte Programm beschäftigt sich mit dem so genannten „Acht-Damen-Problem“.



Die nachfolgende Grafik zeigt die ersten drei Lösungen (von insgesamt 92).



```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<TITLE> Acht-Damen-Problem </TITLE>
<META CHARSET="ISO-8859-1">
<META NAME="viewport" CONTENT="width=device-width, initial-scale=1.0">
<META NAME="description" CONTENT=" Acht-Damen-Problem ">
<META NAME="author" CONTENT="Ralf Pfeifer & Herbert Paukert">

 <SCRIPT>

    var count = 0;  // Zähler aller gefundenen Lösungen
    var info = '';  // Textliste aller gefundenen Lösungen

    // Hintergrundfarbe und Inhalt der einzelnen Felder des Schachbretts bestimmen
    // Inhalt des Arrays 'TabellenZelle' deklarieren

    var TabellenZelle = new Array(4);
        TabellenZelle[0] = "<TD BGCOLOR=white> </TD>";
        TabellenZelle[1] = "<TD BGCOLOR=gray> </TD>";
        TabellenZelle[2] = "<TD BGCOLOR=white><FONT COLOR=gray><B>D</B></FONT> </TD>";
        TabellenZelle[3] = "<TD BGCOLOR=gray><FONT COLOR=white><B>D</B></FONT> </TD>";


    // Prüft, ob die zuletzt positionierte Dame von den anderen geschlagen werden kann

    function DameKannSchlagen() {
      for (var Spalte = LS - 1; Spalte >= 0; Spalte--) {
          // Zwei Damen in der gleichen Zeile ?
          if (SchachBrett[Spalte] == SchachBrett[LS]) { return true; }
          // Zwei Damen auf der gleichen Diagonalen ?
          if (Math.abs(SchachBrett[Spalte] - SchachBrett[LS]) == LS - Spalte) { return true; }
       }
       return false;
    }


    // Ermittelt die nächste Spielstellung

    function NeueStellung() {
        // Dame ein Feld nach oben verschieben.
        // Falls Brettrand erreicht, letzte Dame verschieben
        while (((++SchachBrett[LS]) >= BrettGroesse) && (LS > 0)) { LS--; }
        // Alle Stellungen durchprobiert ?
        Weiter = ((SchachBrett[0] != BrettGroesse) || (LS != 0));
    }


    // Spielbrett ausgeben

    function AusgabeInHTML() {
        var Spalte, Tabelle;
        Tabelle  = "<P><P><HR><TABLE BORDER=2 CELLPADDING=4 CELLSPACING=2>";
        Tabelle += "<CAPTION>Stellung Nr. " + AnzLoesungen + "</CAPTION>";

        for (var Zeile = 0; Zeile < BrettGroesse; Zeile++) {
           Tabelle += "<TR>";

           for (Spalte = 0; Spalte < BrettGroesse; Spalte++) {
               FeldMuster  = (SchachBrett[Zeile] == Spalte) ? 2 : 0;
               FeldMuster += (Zeile + Spalte) % 2;
               Tabelle    += TabellenZelle[FeldMuster];
               s = TabellenZelle[FeldMuster];

               if (s.indexOf('>D<') > 0 ) {
                  info = info + (Spalte+1).toString();
               }
           }
           Tabelle += "</TR>";
        }

        info = info + '\n';
        Seite.writeln(Tabelle + "</TABLE><P><P>");
    }
```

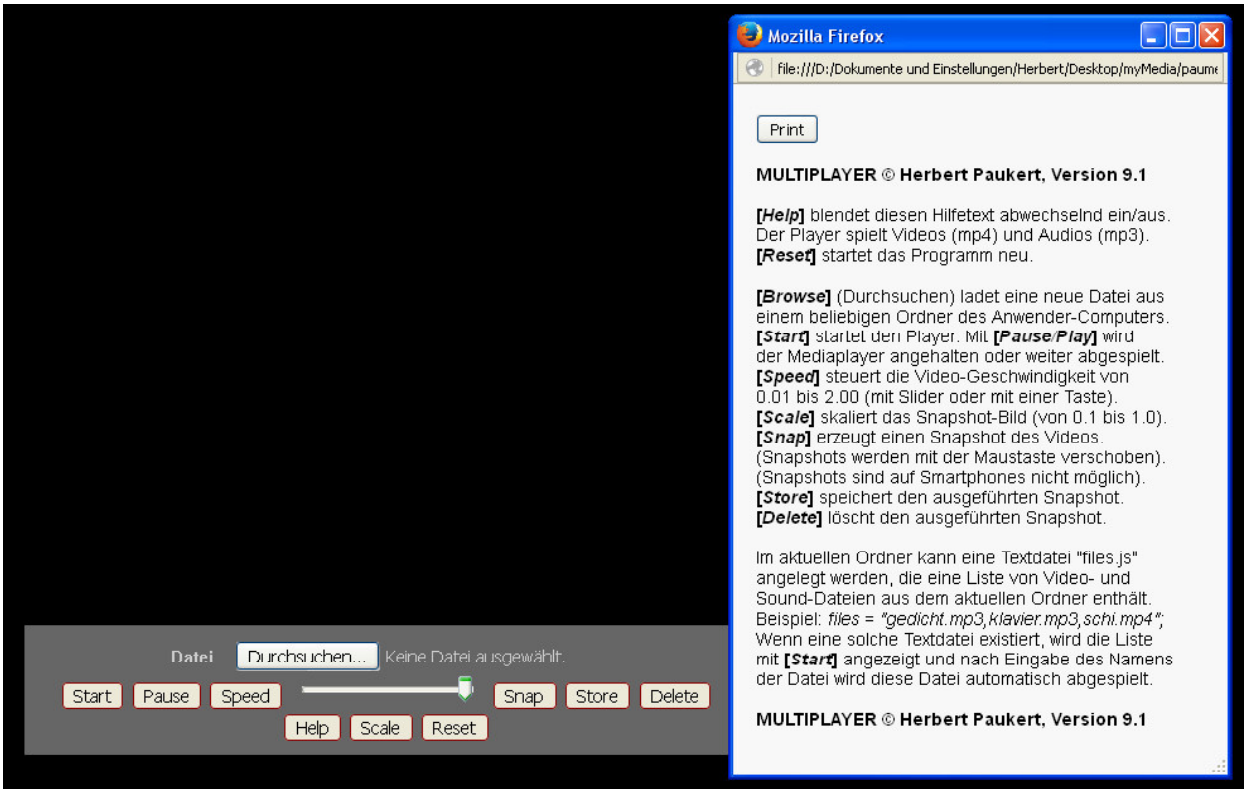
```
    // Hauptprogramm starten

    function Start(EingabeFeld) {

        // Variable vorbelegen
        info = '';
        count = 0;
        BrettGroesse = EingabeFeld.value;
        SchachBrett = new Array(BrettGroesse);
        LS = 1;                  // Letzte Spalte
        AnzLoesungen = 0;        // Anzahl gefundener Loesungen
        Weiter = true;           // Steuervariable

        NeuesFenster = open("","","scrollbars,resizable,menubar")
        Seite = NeuesFenster.document;

        // Brett aufräumen
        for(var i = 0; i < BrettGroesse; )  { SchachBrett[i++] = 0; }

        // Neues Browser-Fenster für die Ausgabe öffnen
        // Positionen ausprobieren
        while (Weiter) {
            while (DameKannSchlagen()) { NeueStellung(); }
            if (LS == (BrettGroesse - 1)) {
                 // Eine Lösung gefunden
                 AnzLoesungen++;
                 AusgabeInHTML();
                 NeueStellung();
            }
            else { SchachBrett[++LS] = 0; }
        }

        // Anzahl der gefundenen Stellungen ausgeben
        count = AnzLoesungen;
        document.getElementById("ANZ").innerHTML = "Lösungs-Anzahl = " + count;
        document.getElementById("TE").value = info;

        // Eingabe auf der HTML-Seite beenden
        NeuesFenster.focus();
        Seite.close();
    }
</SCRIPT>

<STYLE>
   BODY { margin:50px; background-color:#DDDDEE; font-family:Arial }
   TABLE, TR, TH, TD { vertical-align:top }
   #ANZ { position: absolute; left:600px; top:10px; }
   #TE  { position: absolute; left:600px; top:50px; border: 2px solid #666666; padding:10px;
     font-family:Courier-New; font-size: 14px; font-weight:bold;  }
</STYLE>

</HEAD>


<BODY>
<H2>Das Acht-Damen-Problem</H2>

<P>Acht Damen sind auf einem Schachbrett so zu positionieren,<BR>
daß keine die andere schlagen kann!
</P>

<P>Für die Damen gelten für das Schlagen die üblichen Schachregeln:</P>

<UL>
   <LI>Die Dame kann auf den Diagonalen schlagen </LI>
   <LI>Die Dame kann auf der Horizontalen und Vertikalen schlagen </LI>
   <LI>Die Dame kann über jede Entfernung schlagen </LI>
</UL>

<P>Das folgende Formular zeigt alle Lösungen des Problems.<BR>
Es muss dann nach der Öffnung wieder geschlossen werden ......<BR>
Hinweis: Das ursprüngliche Programm stammt von "Ralf Pfeifer".<BR>
Es wurde 2019 von "Herbert Paukert" modfifiziert.
</P>
```

```
<P ID="ANZ"> Lösungs-Anzahl = 0 </P>

<TEXTAREA ID="TE" COLS="20" ROWS="25">
</TEXTAREA>

<FORM>
  <CENTER><TABLE BORDER="1" CELLPADDING="5" ALIGN="left">
    <TR>
      <TD>Brettgröße</TD>
      <TD><INPUT NAME="BrettGr" TYPE="number" VALUE="8" SIZE="3"></TD>
      <TD><INPUT TYPE="button" VALUE="Lösungen ermitteln"
                               ONCLICK="Start(this.form.BrettGr)"></TD>
    </TR>
  </TABLE>
  </CENTER>
</FORM>

</TABLE>

<BR><BR><BR>
<P>
Am Programmende stehen in der rechten Textbox<BR>
alle gefundenen Lösungen. Die Ziffern einer Lösung<BR>
sind die Spaltenpositionen der Damen in den Zeilen<BR>
des Schachbrettes. Beispielsweise bedeutet "15863724":<BR>
Dame in Zeile 1 / Spalte 1, Dame in Zeile 2 /Spalte 5,<BR>
Dame in Zeile 3 / Spalte 8, usw.
</P>

</BODY>
</HTML>
```

Programmieren mit JavaScript, Teil 7
Multimedia und Bildverarbeitung

(7.1.1) Ein universeller Mediaplayer

Das erste Programm „**paumedia.html**“ von Teil 7 ist ein universeller Mediaplayer, dessen grundsätzliche Bedienung in der folgenden Grafik beschrieben ist.



Snapshot (image.jpg) aus dem Schi-Video (schifahr.mp4).

Erklärungen

In der Hilfe des Mediaplayers wird seine Bedienung erläutert. Hervorzuheben sind folgende Funktionen:

(1) Die Funktion „**fileSelect(event)**“ ist ein Ereignishandler für das HTML-Inputobjekt „**file**“:

```
<input type="file" id="files" name="files[]" accept=".mp3,.wav,.mp4,.webm">
document.getElementById("files").addEventListener("change",fileSelect,false);.
```

Damit wird zuerst eine Mediadatei in beliebigen Ordnern des lokalen Computers gesucht und als erste Datei „**files[0]**“ der Dateiliste „**files[]**“ erfasst. Mit „**Filereader**“ wird sie dann gelesen und in ein Blob-Objekt übertragen. Nach Ermittlung des lokalen Dateipfades „**url**“ wird dieser als Videoquelle verwendet und die Mediadatei mit der Funktion „**videoStart()**“ im Mediaplayer geöffnet.

(2) Alternativ kann nur mit der Funktion „**videoStart()**“ alleine eine Mediadatei, die sich im aktuellen Ordner des Computers befindet, direkt geöffnet werden. Das ist nur dann möglich, wenn im Ordner eine Textdatei „**myfiles.js**“ angelegt ist, in welcher eine Liste von Mediadateien (mp3, mp4) steht. Diese wird angezeigt und man kann daraus eine Datei direkt wählen und im Mediaplayer öffnen. Zur bequemen Handhabung genügt dabei die Eingabe der Dateinummer aus der angezeigten Liste.

(z.B. Inhalt von „**myfiles.js**“: *files = "gedicht.mp3,musik.mp3,donald.mp4,schifahr.mp4";*)

(3) Die Funktion „**slideSpeed()**“ ist ein Slider-Objekt mit welchem die Geschwindigkeit der Abspielung der Mediadatei im Player gesteuert wird. Dabei wird die „**playbackRate**“ des Players verwendet. Zusätzlich kann die Geschwindigkeit auch händisch mit der Funktion „**setSpeed()**“ eingegeben werden. Der Geschwindigkeitsbereich erstreckt sich von der Superzeitlupe (0.01) bis zum doppelten Normalwert (2.0). Bei normaler Geschwindigkeit ist die „**playbackRate**“ genau 1.0.

(4) Das Programm ermöglicht auch die Erzeugung von Standbildern (**snapshots**) aus dem Videostrom. Dazu wird zuerst im body-Abschnitt ein Canvas-Objekt angelegt. Mit der Funktion „**snapShot()**“ wird auf dieses Canvas-Objekt ein entsprechendes Video-Standbild abgebildet. Zusätzlich kann dann mit der Funktion „**storeImage()**“ mithilfe eines Links das Bild in den Download-Ordner als eine jpg-Grafikdatei abgespeichert werden. Die Funktion „**delImage()**“ löscht den Schnappschuss. Mit der Funktion „**snapScale()**“ kann die Bildgröße des Schnappschusses eingestellt werden.

(5) Mithilfe der Funktion „**dragElement(elem)**“ kann das Schnappschuss-Bild bei gehaltener Maustaste am Bildschirm beliebig verschoben werden.

Programmlisting

```
<!DOCTYPE html>
<html>
<head>
<title>Mediaplayer</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="Description" content="Universeller Mediaplayer für Videos und Audios">
<meta name="author" content="Herbert Paukert">
<style>
  body { font-family:System,sans-serif; font-size: 20px; background:#000; }
  #abs0 { visibility: hidden; }
  #files { font-size: 18px; margin: 4px; }
  #videoBox { width:640px; background:#000; margin:auto; margin-top:40px;   }
  #videoBar { background:#666; padding:10px; color:#CCC; text-align:center; }
  #slider { width: 150px; }
  #help { color:#FFF; visibility: hidden; text-align:center;}
  #lnk { color:yellow; }
  .btn { font-size: 20px; border: 2px solid darkred; border-radius: 4px; margin: 2px; }
</style>

<script src = "myfiles.js"></script>
```

```
<script>
  // paumedia.html, Version 9.1
  var childWindow;
  var child = false;
  window.addEventListener( "unload", function() { closeChild(); } );

  function closeChild() {
  // Hilfetext schließen
     if (!childWindow || childWindow.closed) { return; }
     if (child) { childWindow.document.close(); childWindow.close(); }
  }

  function showHelp() {
  // Hilfetext anzeigen
     if (child) { closeChild(); child = false; return; }
     child = true;
     childWindow = window.open('','childWindow','top=20, left=970, width=430, height=580,
          srollbars=yes, menubar=no, toolbar=no');
     childWindow.document.open();
     childWindow.document.write('<html><head><meta name="viewport" content="width=device-width,
          initial-scale=1.0"></head>');
     childWindow.document.write('<body style="background-color:#F8F8F8; font-family: sans-serif;
          font-size:14px; text-size-adjust:none; padding-left:10px; padding-bottom:10px;">');
     childWindow.document.write('<br><input type="button" value="Print"
          onclick="print();"><br><br>');
     childWindow.document.write('<b>MULTIPLAYER &copy Herbert Paukert, Version 9.1 </b><br><br>');
     childWindow.document.write('<b>[<i>Help</i>]</b> blendet diesen Hilfetext abwechselnd
          ein/aus.<br>');
     childWindow.document.write('Der Player spielt Videos (mp4) und Audios (mp3).<br>');
     childWindow.document.write('<b>[<i>Reset</i>]</b> startet das Programm neu.<br><br>');
     childWindow.document.write('<b>[<i>Browse</i>]</b> (Durchsuchen) ladet eine neue Datei
          aus<br>');
     childWindow.document.write('einem beliebigen Ordner des Anwender-Computers.<br>');
     childWindow.document.write('<b>[<i>Start</i>]</b> startet den Player. Mit
          <b>[<i>Pause/Play</i>]</b> wird<br>');
     childWindow.document.write('der Mediaplayer angehalten oder weiter abgespielt.<br>');
     childWindow.document.write('<b>[<i>Speed</i>]</b> steuert die Video-Geschwindigkeit von<br>');
     childWindow.document.write('0.01 bis 2.00 (mit Slider oder mit einer Taste).<br>');
     childWindow.document.write('<b>[<i>Scale</i>]</b> skaliert das Snapshot-Bild (von 0.1 bis
          1.0).<br>');
     childWindow.document.write('<b>[<i>Snap</i>]</b> erzeugt einen Snapshot des Videos.<br>');
     childWindow.document.write('(Snapshots werden mit der Maustaste verschoben).<br>');
     childWindow.document.write('(Snapshots sind auf Smartphones nicht möglich).<br>');
     childWindow.document.write('<b>[<i>Store</i>]</b> speichert den ausgeführten Snapshot.<br>');
     childWindow.document.write('<b>[<i>Delete</i>]</b> löscht den ausgeführten Snapshot.<br><br>');
     childWindow.document.write('Im aktuellen Ordner kann eine Textdatei &quot;files.js&quot;<br>');
     childWindow.document.write('angelegt werden, die eine Liste von Video- und<br>');
     childWindow.document.write('Sound-Dateien aus dem aktuellen Ordner enthält.<br>');
     childWindow.document.write('Beispiel: <i>files = &quot; gedicht.mp3, klavier.mp3,
          schi.mp4&quot;;</i><br>');
     childWindow.document.write('Wenn eine solche Textdatei existiert, wird die Liste<br>');
     childWindow.document.write('mit <b>[<i>Start</i>]</b> angezeigt und nach Eingabe des
          Namens<br>');
     childWindow.document.write('der Datei wird diese Datei automatisch abgespielt.<br><br>');
     childWindow.document.write('<b>MULTIPLAYER &copy Herbert Paukert, Version 9.1 </b><br>');
     childWindow.document.write('</body></html>');
  }

  var dat;                    // Datei-Variable
  var datName;                // Dateiname
  var canvas;                 // Canvas-Variable
  var video;                  // Video-Variable
  var vwidth = 640;           // Video-Breite
  var url;                    // Adressenzeiger bzw. Pfad
  var speed  = 1.0;           // Video-Speed
  var fscale = 0.5;           // Snapshot-Skalierung
  var select = false;         // Kennvariable für Video-Auswahl
  var start  = false;         // Kennvariable für Video-Start
  var snap   = false;         // Kennvariable für Snapshot-Erzeugung
  var store  = false;         // Kennvariable für Snapshot-Speicherung
  var anchor;                 // Adressenzeiger bzw. Pfad
  var imageName = 'image.jpg';  // Bildname (ev. auch 'image.png')
  var smallScr = false;       // Kennvariable für Smallscreens von Smartphones

  function absatz(vater,kind) {
  // Erzeugt einen neuen Absatz “kind” unter dem Element “vater”
     var parent = document.getElementById(vater);
     var abs1 = document.createElement("p");
     abs1.setAttribute("id",kind);
     var txt = document.createTextNode("HP");
     abs1.appendChild(txt);
     parent.appendChild(abs1);
  }
```

```
function initPlayer(){
  // Initialisierungen
   if (screen.width < 800) { smallScr = true; }
   else  { smallScr = false; }
   if (smallScr) { alert(' Bild auf Minimum verkleinern \n und dann ein weiter ..... '); };
   if ((screen.width >= 800) && (screen.width <= 1500)) {
     document.getElementById("body").style.fontSize = "16px";
     document.getElementById("files").style.fontSize = "16px";
     var liste = document.getElementsByClassName("btn");
     for(var i = 0; i < liste.length; i++) { liste[i].style.fontSize = "16px"; }
   }
   if (screen.width > 1500) {
     vwidth = 800;
     document.getElementById("videoBox").style.width = vwidth +"px";
     document.getElementById("myVideo").style.width = vwidth;
   }
   if (screen.width > 1800) {
     absatz("abs0","abs1");
     absatz("abs0","abs2");
   }
  // set object references
    dat = document.getElementById("files");
    body = document.getElementById('body');
    canvas = document.getElementById('myCanvas');
    video  = document.getElementById("myVideo");
    vidBox = document.getElementById("videoBox");
    vidBar = document.getElementById("videoBar");
    vidbtn = document.getElementById("videobtn");
    playbtn = document.getElementById("playbtn");
    speedbtn = document.getElementById("speedbtn");
    sslider = document.getElementById("slider");
    scalebtn = document.getElementById("scalebtn");
    snapbtn = document.getElementById("snapbtn");
    storebtn = document.getElementById("storebtn");
    delbtn.addEventListener("click",delImage,false);
    helpbtn = document.getElementById("helpbtn");
    resetbtn = document.getElementById("resetbtn");
  // add event listeners
    dat.addEventListener("change",fileSelect,false);
    body.addEventListener("keydown",setSpeed,false);
    vidbtn.addEventListener("click",videoStart,false);
    speedbtn.addEventListener("click",setSpeed,false);
    sslider.addEventListener("change",slideSpeed,false);
    playbtn.addEventListener("click",playPause,false);
    scalebtn.addEventListener("click",snapScale,false);
    snapbtn.addEventListener("click",snapShot,false);
    storebtn.addEventListener("click",storeImage,false);
    delbtn.addEventListener("click",delImage,false);
    helpbtn.addEventListener("click",showHelp,false);
    resetbtn.addEventListener("click",reset,false);
  }

  function fileSelect(event) {
  // select "first file" from "files[]"
    select = true;
    var file = event.target.files[0];
    datname = file.name;
    var reader = new FileReader();
    reader.readAsArrayBuffer(file);
    reader.onload = function() {
      let buffer = reader.result;
      let blob = new Blob([new Uint8Array(buffer)]);
      url = window.URL.createObjectURL(blob);
    };
  }

  function videoStart() {
  // start video
    if (files && !select ) {
      var arr = files.split(',');
      var max = arr.length;
      var info = '';
      for (n = 0; n < max; n++) { info = info + (n+1) + ':  ' + arr[n] + '\n'; }
      info = info + '-------------------' +
             '\nDateinummer (1 bis ' + max +
             ') oder\nDateinamen hier eingeben:\n ';
      var s = prompt(info,'1').trim();
      var n = 1 * s - 1;
      if ((!isNaN(n)) || (n >= 0) || (n <= max)) { datName = arr[n]; }
      else { datName = s; }
      video.src = datName;
    }
```

```
else { video.src = url; }
   start = true;
   select = false;
   video.width = vwidth;
   video.controls = true;
   video.playbackRate = speed;
   video.play();
   playbtn.innerHTML = "Pause";
 }

 function slideSpeed() {
 // slide videospeed
   var x = sslider.value;
   speed = x / 100;
   video.playbackRate = speed;
 }

 function setSpeed() {
 // set videospeed
   sslider.value = 100;
   var s = prompt('Speed (0.01 - 2.0)','1.0');
   speed = 1 * s;
   if ( (speed < 0.01) || (speed > 2.0) ) { speed = 1.0; }
   sslider.value = 100*speed;
   video.playbackRate = speed;
 }

 function playPause() {
 // play or pause
   if (!start) { alert('Zuerst [Datei auswählen] bzw. [Start] anklicken!'); return; }
   video.controls = true;
   if (video.paused) { video.play(); playbtn.innerHTML = "Pause"; }
   else { video.pause(); playbtn.innerHTML = "Play"; }
 }

 function reset() {
 // reset
   fscale = 0.5;
   speed = 1.0;
   sslider.value = 100;
   dat.value = "";
   window.scrollTo(0,0);
   window.location.reload();
 }

 function getposX(elem) {
 // X-Position ermitteln
   var elemData = document.getElementById(elem).getBoundingClientRect();
   return elemData.left;
 }

 function getposY(elem) {
 // Y-Position ermitteln
   var elemData = document.getElementById(elem).getBoundingClientRect();
   return elemData.top;
 }

 function setposXY(elem,posX,posY) {
 // X-Position und Y-Position setzen
   X = parseInt(getposX(elem));
   Y = parseInt(getposY(elem));
   var el = document.getElementById(elem);
   el.style.left = (el.offsetLeft - X) + posX + "px";
   el.style.top  = (el.offsetTop - Y) + posY + "px";
 }

 function snapScale() {
 // scale Snapshot
   if (smallScr) { return; }
   var t = fscale.toString();
   var s = prompt('Snapshot-Skalierung (0.1 - 1.0)',t);
   fscale = 1 * s;
   if ( (fscale < 0.1) || (fscale > 1.0) ) { fscale = 1.0; }
 }

 function snapShot() {
 // Snapshot erzeugen
   if (smallScr) { return; }
   if (!start) { alert('Zuerst [Datei auswählen] bzw. [Start] anklicken!'); return; }
   if (store)  { document.body.removeChild(anchor); store = false; }
   canvas.width  = parseInt(video.videoWidth * fscale);
   canvas.height = parseInt(video.videoHeight * fscale);
```

```
    var ctx = canvas.getContext("2d");
    ctx.drawImage(video, 0, 0, canvas.width, canvas.height);
    canvas.style.border = "1px solid black";
    setposXY("pic",10,10);
    snap = true;
  }

  function storeImage() {
  // Snapshot abspeichern
    if (smallScr) { return; }
    if (!start) { alert('Zuerst [Datei auswählen] bzw. [Start] anklicken!'); return; }
    if (!snap) { alert('Zuerst [Snap] anklicken!'); return; }
    anchor = document.createElement("a");
    anchor.download = imageName;
    if (imageName.indexOf('png') > -1) { anchor.href = canvas.toDataURL("image/png"); }
    if (imageName.indexOf('jpg') > -1) { anchor.href = canvas.toDataURL("image/jpeg"); }
    anchor.innerHTML = "  download image  ";
    document.body.appendChild(anchor);
    anchor.setAttribute("id","lnk");
    anchor.click;
    store = true;
  }

  function delImage() {
  // Snapshot löschen
    if (smallScr) { return; }
    if (!start) { alert('Zuerst [Datei auswählen] bzw. [Start] anklicken!'); return; }
    if (!snap)  { alert('Zuerst [Snap] anklicken!'); return; }
    if (store)  { document.body.removeChild(anchor); store = false; }
    canvas.width  = 0;
    canvas.height = 0;
    snap = false;
  }

  function dragElement(elem) {
   // Hauptprogramm mit drei Unterprogrammen
     var pos1 = 0, pos2 = 0, pos3 = 0, pos4 = 0;
     elem.onmousedown = dragMouseDown;

     function dragMouseDown(ev) {
     // Erstes Unterprogramm
       ev.preventDefault();
       pos3 = ev.clientX;
       pos4 = ev.clientY;
       document.onmousemove = elementDrag;
       document.onmouseup = closeDragElement;
     }

     function elementDrag(ev) {
     // Zweites Unterprogramm
       ev.preventDefault();
       pos1 = pos3 - ev.clientX;
       pos2 = pos4 - ev.clientY;
       pos3 = ev.clientX;
       pos4 = ev.clientY;
       elem.style.top = (elem.offsetTop - pos2) + "px";
       elem.style.left = (elem.offsetLeft - pos1) + "px";
     }

     function closeDragElement() {
     // Drittes Unterprogramm
       document.onmousemove = null;
       document.onmouseup = null;
     }
  }

  window.onload = initPlayer;
  document.onunload = closeChild;

</script>
</head>

<body id="body">
<br>
<p id="abs0"> </p>
<div id="videoBox">
  <video id="myVideo" width="640" height="480">
    <source src="" controls="true">
  </video>
  <div id="videoBar">
    Datei 
    <input type="file" id="files" name="files[]" accept=".mp3,.wav,.mp4,.webm">
    <br>
```

```
    <button id="videobtn" class="btn">Start</button>
    <button id="playbtn" class="btn">Pause</button>
    <button id="speedbtn" class="btn">Speed</button>
    <input  id="slider" type="range" min="0" max="100" value="100" step="1">
    <button id="snapbtn" class="btn">Snap</button>
    <button id="storebtn" class="btn">Store</button>
    <button id="delbtn" class="btn">Delete</button><br>
    <button id="helpbtn" class="btn">Help</button>
    <button id="scalebtn" class="btn">Scale</button>
    <button id="resetbtn" class="btn">Reset</button>
  </div>
</div>

<div id="pic" style="position:absolute; top:0px; left:0px;">
  <canvas id="myCanvas" top = "0" left = "0" width="640" height="480">
</div>

<script>
   dragElement(document.getElementById("pic"));
</script>

<br>
</body>
</html>
```

(7.1.2) Sound-Recording

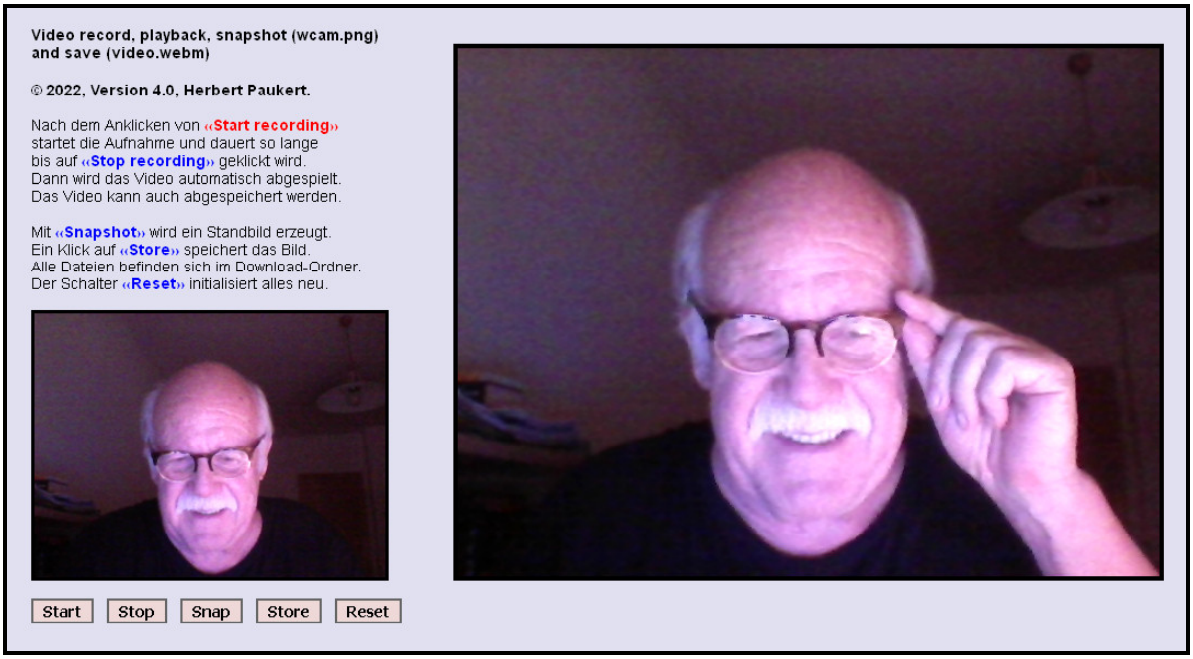
Mit dem Programm „**paumicro.html**“ können Mikrofonaufnahmen ausgeführt und dann optional als „**wav-Dateien**“ im Download-Ordner abgespeichert werden. Ausführliche Erklärungen findet man im nächsten Programm „**paucamera.html**“.

Programmlisting

```
<!DOCTYPE html>
<html>
<head>
<title>Voicerecorder</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Audiorecording">
<meta name="author" content="Paukert Herbert">
<style>
body { background-color:#F0F0D8; color:black;
       font-family:sans-serif,System; font-size:14px; text-align:left; padding:2% }
.cd  { border:2px solid #666666; background-color:#EED8D8; font-size:14px; font-weight: bold; }
a:link { color:red; background-color:#E8E8FF; font-weight:bold; text-decoration:underline; }
</style>
```

```
<script>
// paumicro.html, Version 4.0
var start = false;
navigator.mediaDevices.getUserMedia({audio:true}).then(stream => {
   rec = new MediaRecorder(stream);
   audioData = [];
   rec.ondataavailable = event => {
      audioData.push(event.data);
      if (rec.state == "inactive") {
        let blob = new Blob(audioData,{type:'audio/wav'});
//        let blob = new Blob(audioData,{type:'audio/mp3'});
        recAudio.src = URL.createObjectURL(blob);
        recAudio.controls = true;
        recAudio.autoplay = true;
        createAudioLink(recAudio.src);
      }
   }
}).catch(event => alert(event));

function createAudioLink(blobUrl) {
    let downloadEl = document.createElement('a');
    downloadEl.style = 'display: block';
    downloadEl.innerHTML = 'download';
    downloadEl.download = 'audio.wav';
//    downloadEl.download = 'audio.mp3';
    downloadEl.href = blobUrl;
    document.body.appendChild(downloadEl);
}

function startRecord() {
  start = true;
  show();
  startRecord.disabled = true;
  stopRecord.disabled = false;
  audioData = [];
  rec.start();
}

function stopRecord() {
  if (!start) { alert('Zuerst «Start recording» anklicken!'); return; }
  hide();
  startRecord.disabled = false;
  stopRecord.disabled = true;
  rec.stop();
  start = false;
}

function resetRecord() {
  start = false;
  window.location.reload();
}

function show() { document.getElementById("info").style.visibility = 'visible'; }
function hide() { document.getElementById("info").style.visibility = 'hidden';  }
</script>
</head>

<body onload="hide()" oncontextmenu="return false">
<form>
<h3>
  Sound record, playback and save (audio.wav)
</h3>
  <b> © 2022, Version 4.0, Herbert Paukert.</b><br>
  <br>
  Nach dem Anklicken von <b><font color = 'red'> «Start recording» </font></b><br>
  startet die Aufnahme und dauert so lange<br>
  bis auf<b><font color = 'blue'> «Stop recording» </font></b> geklickt wird.<br>
  Dann wird der Sound automatisch abgespielt.<br><br>
  Der Sound kann auch abgespeichert werden.<br>
  Alle Dateien befinden sich im Download-Ordner.<br>
  Der Schalter <b><font color = 'blue'> «Reset» </font></b> initialisiert alles neu.<br>
<br>
<audio id="recAudio" controls="true"></audio>
<br><br>
<input type="button" id="start" value="Start rec." size="4" class="cd" onclick= "startRecord()">; 
<input type="button" id="stop" value="Stop rec." class="cd" size="4" onclick= "stopRecord()">; 
<input type="button" id="reset" value="Reset" class="cd" size="4" onclick= "resetRecord()">
<br>
<div id="info"><i>Bitte jetzt in das Mikrofon sprechen . . .</i></div>
</form>
</body>
</html>
```

(7.1.3) Video-Recording



Mit dem Programm „**paucamera.html**“ werden Aufnahmen mit der Webkamera ausgeführt.
Die Videos werden dann optional als „**webm-Dateien**“ im Download-Ordner abgespeichert.
Zusätzlich erzeugte Standbilder (snapshots) werden optional als „**png-Dateien**“ gespeichert.
Erklärungen zum Programm findet man am Ende des Programmlistings.

Programmlisting

```
<!DOCTYPE html>
<html>
<head>
<title>videorecorder</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Videorecording">
<meta name="author" content="Paukert Herbert">
<style>
body { background-color:#F0F0D8; color:black;
       font-family:sans-serif,System; font-size:14px; text-align:left; padding:2% }
.cd  { border:2px solid #666666; background-color:#EED8D8; font-size:14px; font-weight: bold; }
#box { width:320px; height:240px; border:3px solid #000000; }
a:link { color:red; background-color:#E8E8FF; font-weight:bold; text-decoration:underline; }
</style>

<script>
// paucamera.html, Version 4.0
var start = false;    // Kennvariable für Videoaufnahmen
var save  = false;    // Kennvariable für Videospeicherung
var snap  = false;    // Kennvariable für Snapshots
var fscale = 0.5;     // Skalierungsvariable für Snapshots

navigator.mediaDevices.getUserMedia({video:true, audio:true}).then(stream => {
   rec = new MediaRecorder(stream);
   recVideo.srcObject = stream;
   videoData = [];
   rec.ondataavailable = event => {
      videoData.push(event.data);
        if (rec.state == "inactive") {
          let blob = new Blob(videoData,{type:'video/webm'});
//        let blob = new Blob(videoData,{type:'video/mp4'});
          recVideo.src = URL.createObjectURL(blob);
          createVideoLink(recVideo.src);
      }
   }
}).catch(event => alert(event));
```

```
function createVideoLink(blobUrl) {
    if (save) { return; }
    downloadEl = document.createElement('a');
    downloadEl.style = 'display: block';
    downloadEl.innerHTML = 'download';
    downloadEl.download = 'video.webm';
//    downloadEl.download = 'video.mp4';
    downloadEl.href = blobUrl;
    downloadEl.innerHTML = "  download video  ";
    document.body.appendChild(downloadEl);
    save = true;
}

function startRecord() {
  if (save) { resetRecord(); }
  start = true;
  show();
  startRecord.disabled = true;
  stopRecord.disabled = false;
  videoData = [];
  rec.start();
}

function stopRecord() {
  if (!start) { alert('Zuerst «Start recording» anklicken!'); return; }
  hide();
  startRecord.disabled = false;
  stopRecord.disabled = true;
  recVideo.controls = true;
  recVideo.muted = false;
  let stream = recVideo.srcObject;
  stream.getTracks().forEach(track => track.stop());
  rec.stop();
  recVideo.srcObject = null;
  start = false;
}

function snapShot() {
  canvas = document.getElementById('myCanvas');
  canvas.width  = recVideo.videoWidth * fscale;
  canvas.height = recVideo.videoHeight * fscale;
  let context2D = canvas.getContext("2d");
  context2D.drawImage(recVideo, 0, 0, canvas.width, canvas.height);
  canvas.style.border = "2px solid black";
  snap = true;
}

function storeImage() {
  if (!snap) { alert('Zuerst «Snapshot» anklicken!'); return; }
  canvas = document.getElementById('myCanvas');
  anchor = document.createElement("a");
  anchor.href = canvas.toDataURL("image/png");
  anchor.download = "wcam.png";
//  anchor.href = canvas.toDataURL("image/jpg");
//  anchor.download  = "wcam.jpg";
  anchor.innerHTML = "  download image  ";
  document.body.appendChild(anchor);
  anchor.click;
}

function resetRecord() {
  start = false;
  save  = false;
  snap  = false;
  window.location.reload();
}

function show() {
  document.getElementById("info").style.visibility = 'visible';
  document.getElementById("box").style.border = "3px solid #FF0000";
}

function hide() {
  document.getElementById("info").style.visibility = 'hidden';
  document.getElementById("box").style.border = "3px solid #000000";
}

</script>
</head>
```

```
<body onload="hide()" oncontextmenu="return false">
<div id="pic" style="position:absolute; top:70px; left:420px;">
  <canvas id="myCanvas" width="320" height="240">
</div>
<h4>
  Video record, playback, snapshot (wcam.png)<br>
  and save (video.webm)
</h4>
  <b> © 2022, Version 4.0, Herbert Paukert.</b><br>
  <br>
  Nach dem Anklicken von <b><font color = 'red'> «Start recording» </font></b><br>
  startet die Aufnahme und dauert so lange<br>
  bis auf <b><font color = 'blue'> «Stop recording» </font></b> geklickt wird.<br>
  Dann wird das Video automatisch abgespielt.<br>
  Das Video kann auch abgespeichert werden.<br><br>
  Mit <b><font color = 'blue'> «Snapshot» </font></b> wird ein Standbild erzeugt.<br>
  Ein Klick auf <b><font color = 'blue'> «Store» </font></b> speichert das Bild.<br>
  Alle Dateien befinden sich im Download-Ordner.<br>
  Der Schalter <b><font color = 'blue'> «Reset» </font></b> initialisiert alles neu.<br>
<br>
<div id="box">
   <video id="recVideo" width="320" height="240" autoplay muted></video>
</div>
<br>
<input type="button" id="start" value=" Start " size="4" class="cd" onclick= "startRecord()"> 
<input type="button" id="stop" value=" Stop " class="cd" size="4" onclick= "stopRecord()"> 
<input type="button" id="snap" value=" Snap " class="cd" size="4" onclick= "snapShot()"> 
<input type="button" id="store" value=" Store " class="cd" size="4" onclick= "storeImage()"> 
<input type="button" id="reset" value=" Reset " class="cd" size="4" onclick= "resetRecord()"> 
<br>
<div id="info"><i>Bitte jetzt in die Kamera blicken . . .</i></div>
</body>
</html>
```

Erklärungen

(1) Die „**getUserMedia**“-Methode am Scriptanfang hat eine asynchrone Promise-Struktur, welche im Falle einer fehlerlosen Unterstützung durch den Browser das Video-Streaming von der Webkamera zum Screen sicherstellt. Dabei wird die „**Media-Recorder-API**“ des Javascript-Interpreters verwendet, welche ein komplexes Video-Objekt mit Eigenschaften und Methoden zur Verfügung stellt (technische Videoparameter, Recording State, stream, Event-Handler und verschiedene Funktionen wie beispielsweise start(), stop(), pause(), etc.).

Die Programmroutine ist gekennzeichnet durch einen then-Zweig und einen catch-Zweig. Im ersten Ast werden die Videodaten mittels Event-Handler empfangen und dann in einem so genannten Blob-Element zusammengefasst. Am Schluss wird noch ein Link erzeugt, mit dessen Hilfe die Daten in eine Videodatei (.webm oder .mp4) im Download-Ordner gespeichert werden. Falls im ersten Ast ein Fehler auftritt, dann wird im zweiten Ast der Fehler abgefangen und eine entsprechende Fehlermeldung ausgegeben. Die Routine enthält drei Arrow-Funktionen. Ohne diese würde sich nachfolgende Struktur mit drei anonymen, selbstauslösenden Funktionen (unterstrichen) ergeben:

```
if (navigator.mediaDevices.getUserMedia) {
   navigator.mediaDevices.getUserMedia({video:true, audio:true}).then ( function(stream) {
      rec = new MediaRecorder(stream);
      recVideo.srcObject = stream;
      videoData = [];
      rec.ondataavailable = function(event) {
          videoData.push(event.data);
          if (rec.state == "inactive") {
             let blob = new Blob(videoData,{type:'video/webm'});
             recVideo.src = URL.createObjectURL(blob);
             createVideoLink(recVideo.src);
          }
      }
   } ).catch( function(event) {
      alert(event);
   } );
}
```

(2) Das Programm ermöglicht auch die Erzeugung von Standbildern (snapshots) aus dem Videostrom. Dazu wird zuerst im body-Abschnitt ein Canvas-Objekt angelegt. In der Schalter-Routine „**snapShot()**“ wird auf dieses Canvas-Objekt ein entsprechendes Video-Standbild abgebildet. Zusätzlich kann dann mit der Schalter-Routine „**storeImage()**“ mithilfe eines Links das Bild in den Download-Ordner als eine Grafikdatei im png- oder jpg-Format abgespeichert werden.

```
function snapShot() {
  canvas = document.getElementById('myCanvas');
  canvas.width  = recVideo.videoWidth;
  canvas.height = recVideo.videoHeight;
  let context2D = canvas.getContext("2d");
  context2D.drawImage(recVideo, 0, 0, canvas.width, canvas.height);
  canvas.style.border = "4px solid black";
  snap = true;
}

function storeImage(draw) {
  if (!snap) { alert('Zuerst «Snapshot» anklicken!'); return; }
  canvas = document.getElementById('myCanvas');
  anchor = document.createElement("a");
  anchor.href = canvas.toDataURL("image/png");
  anchor.download  = "wcam.png";
//  anchor.href = canvas.toDataURL("image/jpg");
//  anchor.download  = "wcam.jpg";
  anchor.innerHTML = "  download image  ";
  document.body.appendChild(anchor);
  anchor.click;
}
```

(3) Die anderen Programmfunktionen sind selbsterklärend. Die Schalter-Routine „**startRecord()**“ startet die Videoaufnahme. Die Schalter-Routine „**stopRecord()**“ beendet die Videoaufnahme. Bemerkenswert ist dabei, dass dann das im Videoplayer aufgenommene Video automatisch im selben Videoplayer auch abgespielt wird. Das Video kann über den vorher erzeugten Link abgespeichert werden.

In der Schalter-Routine „**stopRecord()**“ werden alle Videospuren mit einem einzigen Befehl gestoppt:

```
  stream.getTracks().forEach(track => track.stop());
```

Eine andere Codevariante dafür ist:

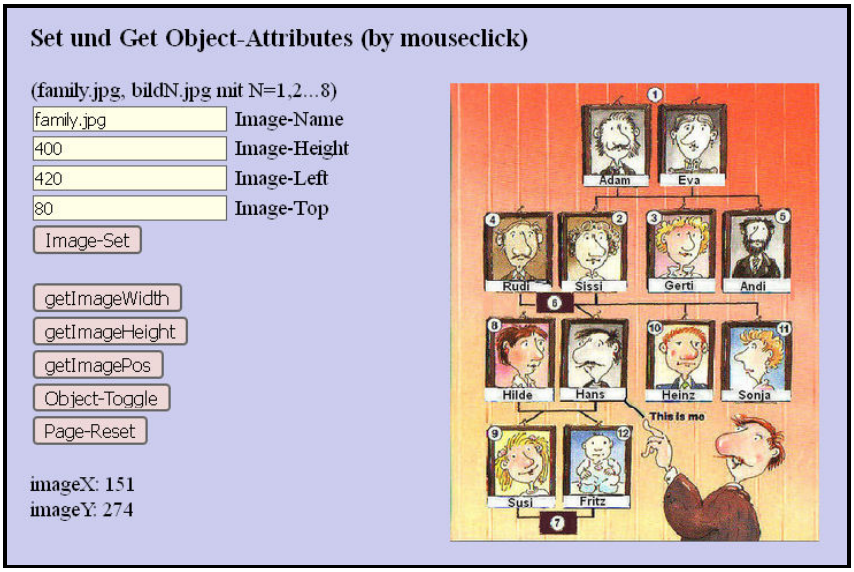
```
  var tracks = stream.getTracks();
  for (var i = 0; i < tracks.length; i++) {
     var track = tracks[i];
     track.stop;
  }
```

Die nachfolgende Funktion `rec.stop();` beendet dann die Videoaufnahme endgültig.

(4) Weitere und ausführlichere Erklärungen zu Promise-Routinen und zum Media-Recorder-API findet man in einschlägigen Dokumentationen im Internet.

Damit ist die Einführung in Media-Playing und Media-Recording abgeschlossen. Auf den nachfolgenden Seiten werden einige wichtige Routinen zur Verarbeitung von Images dargestellt.

(7.2.1) Set/Get von Image-Attributen



Programmlisting

```
<!DOCTYPE html>
<html>
<head>
<title>imageset</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Set/Get von Objektattributen">
<meta name="author" content="Paukert Herbert">

<style>
body   { background-color: #CCCCEE; margin: 2%; padding-left:2%; font-size: 19px; }
#myImg { position: absolute; left: 420px; top: 80px; }
.ein   { margin: 2px; border: 1px solid #666666; background-color: #FFFFE8; font-size: 15px;}
.btn   { margin: 2px; border: 2px solid #666666; background-color: #EED8D8; font-size: 16px;
         border-radius: 4px;}
</style>

<script>
var imgX, imgY;     // Bildpunkt-Koordinaten
var look = false;   // Sperrvariable für Objekt-Identifikation

document.onclick = function(event) {
// Identifikation eines angeklickten Objektes
   if (!look) { return; }
   let ide = event.target.id;
   let aText = document.getElementById(ide).value;
   alert( ide + ": " + aText.toString() );
}

function setPosition(oElement,x,y) {
// Position eines Objektes setzen
   let el = document.getElementById(oElement);
   el.style.position = "absolute";
   el.style.left = x + "px";
   el.style.top =  y + "px";
}

function setImage() {
// Eigenschaften des Bild-Objektes setzen
   document.getElementById("imageX").innerHTML = '';
   document.getElementById("imageY").innerHTML = '';
   let imgName = document.getElementById("inp1").value;
   let imgHeight = document.getElementById("inp2").value;
   let imgLeft = document.getElementById("inp3").value;
   let imgTop = document.getElementById("inp4").value;
   document.getElementById("myImg").src = imgName;
   document.getElementById("myImg").height = imgHeight;
   setPosition('myImg',imgLeft,imgTop);
}
```

```
function getPosition(oElement) {
// Position eines Objektes ermitteln (Variante 1)
  var el = document.getElementById(oElement);
  var xPos = 0;
  var yPos = 0;
  while (el) {
    if ((el.tagName == "body") || (el.tagName == "BODY")) {
      var xScroll = el.scrollLeft || document.documentElement.scrollLeft;
      var yScroll = el.scrollTop || document.documentElement.scrollTop;
      xPos += (el.offsetLeft - xScroll + el.clientLeft);
      yPos += (el.offsetTop  - yScroll + el.clientTop);
    }
    else {
      xPos += (el.offsetLeft - el.scrollLeft + el.clientLeft);
      yPos += (el.offsetTop  - el.scrollTop + el.clientTop);
    }
    el = el.offsetParent;
  }
  let position = new Object();
  position.x = xPos;
  position.y = yPos;
  return position;
}

function getImagePos() {
// Position des Bild-Objektes ermitteln
// mithilfe der Funktion "getPosition"
   let posi = getPosition('myImg')
   let xPos = posi.x;
   let yPos = posi.y;
   document.getElementById("inp3").value = xPos;
   document.getElementById("inp4").value = yPos;
   alert(xPos + ' / ' + yPos);
}

function FindPosition(oElement) {
// Position eines Objektes ermitteln (Variante 2)
  if (typeof( oElement.offsetParent ) != "undefined") {
     for(var posX = 0, posY = 0; oElement; oElement = oElement.offsetParent) {
        posX += oElement.offsetLeft;
        posY += oElement.offsetTop;
     }
     return [posX, posY];
  }
  else { return [oElement.x, oElement.y]; }
}

function GetCoordinates(ev,objEl) {
// Koordinaten eines angeklickten Bildpunktes ermitteln
// mithilfe der Funktion "FindPosition"
  let PosX = 0;
  let PosY = 0;
  let ImgPos = FindPosition(objEl);
  if (!ev) { var ev = window.event; }
  if (ev.pageX || ev.pageY) {
     PosX = ev.pageX;
     PosY = ev.pageY;
  }
  else if (ev.clientX || ev.clientY) {
      PosX = ev.clientX + document.body.scrollLeft
             + document.documentElement.scrollLeft;
      PosY = ev.clientY + document.body.scrollTop
             + document.documentElement.scrollTop;
  }
  PosX = PosX - ImgPos[0];
  PosY = PosY - ImgPos[1];
  imgX = PosX;
  imgY = PosY;
}

function toggle() {
// Wechselschalter für Objekt-Identifikation
   look = !look;
   if (look) { alert('Objekt-Identifikation EINgeschaltet !'); }
   else { alert('Objekt-Identifikation AUSgeschaltet !'); }
}

function fNull(z) {
// Führende Nullen setzen
  z = (z < 10 ? '0' : '') + z;
  z = (z < 100 ? '0' : '') + z;
  return z;
}
```

```
function getImagePoint(event) {
// Koordinaten eines angeklickten Bildpunktes anzeigen
   let myImg = document.getElementById("myImg");
   GetCoordinates(event,myImg);
   imgX = fNull(imgX);
   imgY = fNull(imgY);
   document.getElementById("imageX").innerHTML = imgX;
   document.getElementById("imageY").innerHTML = imgY;
}

function getImageWidth() {
// Bildbreite anzeigen
   let myImg = document.getElementById("myImg");
   let imgWidth = myImg.width;
   alert(imgWidth);
}

function getImageHeight() {
// Bildhöhe anzeigen
   let myImg = document.getElementById("myImg");
   let imgHeight = myImg.height;
   alert(imgHeight);
}

function refresh() {
// Page-Reset
  look = false;
  document.getElementById("inp1").value = "family.jpg";
  document.getElementById("inp2").value = "400";
  document.getElementById("inp3").value = "50";
  document.getElementById("inp4").value = "500";
  window.location.reload();
}

function initPage() {
// Initialisierungen
  document.getElementById("myImg").style.left = "50px";
  document.getElementById("myImg").style.top = "500px";
}

</script>
</head>

<body id="bd" onload="initPage()">
<h3> Set und Get Object-Attributes (by mouseclick) </h3>
<span id="info">(family.jpg, bildN.jpg mit N=1,2...8)</span>
<br>
<input id="inp1" class="ein" type="text" value="family.jpg"> Image-Name<br>
<input id="inp2" class="ein" type="text" value="400"> Image-Height<br>
<input id="inp3" class="ein" type="text" value="420"> Image-Left<br>
<input id="inp4" class="ein" type="text" value="80"> Image-Top<br>
<button id="btn1" class="btn" onclick = "setImage()">Image-Set</button><br>
<br>
<button id="btn2" class="btn" onclick = "getImageWidth()">getImageWidth</button><br>
<button id="btn3" class="btn" onclick = "getImageHeight()">getImageHeight</button><br>
<button id="btn4" class="btn" onclick = "getImagePos()">getImagePos</button><br>
<button id="btn5" class="btn" onclick = "toggle()">Object-Toggle</button><br>
<button id="btn6" class="btn" onclick = "refresh()">Page-Reset</button><br>
<br>
imageX: <span id="imageX"></span>
<br>
imageY: <span id="imageY"></span>
<br>
<img id="myImg" src='family.jpg' height='400' onclick='getImagePoint(event)'>
<br>
</body>
</html>
```

Erklärungen

Der **Event-Handler** am Anfang des Skripts identifiziert ein angeklicktes Objekt und liefert zusätzlich, falls vorhanden, seinen Textinhalt. Der Event-Handler kann mit dem Wechsel-Schalter **[Object-Toggle]** ein- und ausgeschaltet werden.

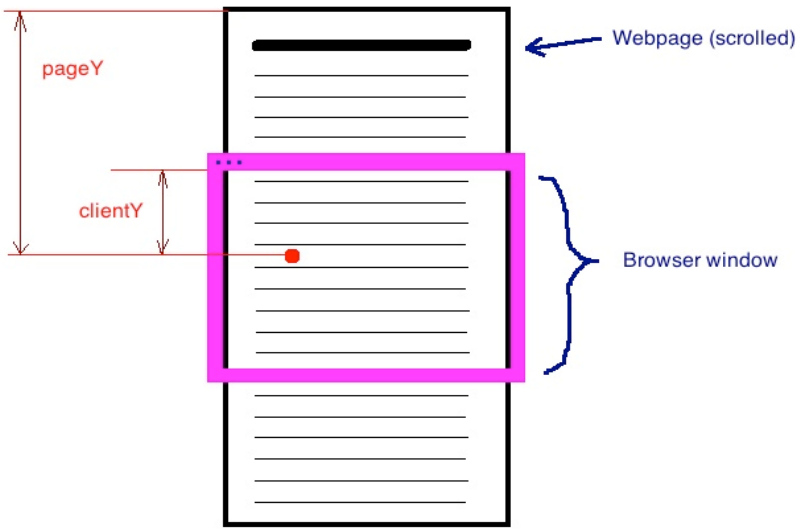
Die Funktion „**setPosition(oElement,x,y)**“ setzt ein Objekt an eine bestimmte Position.

Die Funktion „**setImage()**“ bestimmt ein Bild-Objekt und setzt dessen Position und auch dessen Bildhöhe. Dabei wird die Bildbreite automatisch entsprechend dem originalen Seitenverhältnis (aspect-ratio) eingestellt..

Die Funktion „**getPosition(oElement)**“ ermittelt die absolute Position eines Objektes. Dabei wird die Menge aller DOM-Objekte, beginnend mit dem Wurzelknoten BODY, so lange durchlaufen, bis das Objekt gefunden wird. In diesem Fall werden dann seine absoluten Koordinaten ermittelt. Diese werden mit den entsprechenden Koordinaten aller Eltern-Knoten berechnet. Zusätzlich muss noch ein mögliches Scrollen mitgerechnet werden. Die verwendeten Objektwerte sind „el.scrollLeft“, „el.scrollTop“, „el.offsetLeft“, „el.offsetTop“, „el.clientLeft“ und „el.clientTop“. Die nachfolgende Grafik demonstriert anschaulich diesen Sachverhalt.

Die Funktion „**getImagePos()**“ verwendet die vorangehende Funktion zur Ermittlung der Position des vorliegenden Bildes. Die Koordinaten werden in einem Meldungsfenster und in zwei Input-Feldern ausgegeben.

Die Grafik zeigt ein Browser-Fenster (window) hinter dem eine Webseite (document) scrollt.



(pageX/pageY) beziehen sich absolut auf den linken, oberen Punkt der Webseite.
(clientX/clientY) beziehen sich relativ auf den linken, oberen Punkt des Browser-Fensters (viewport).

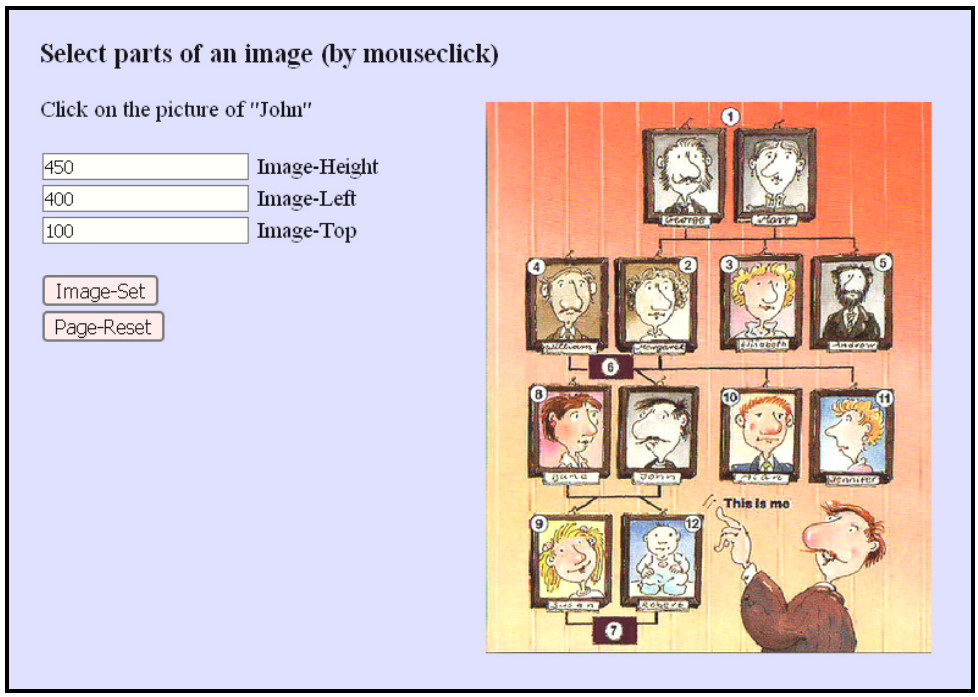
(screenX/screenY) hingegen beziehen sich immer auf den physischen Bildschirm und sind daher abhängig von seiner Auflösung (resolution).

Die Funktion „**getImagePoint(event)**“ ist als Event-Handler dem vorliegenden Bild zugeordnet. Sie verwendet intern die Funktion „**GetCoordinates(ev,objEl)**“, welche ihrerseits die Funktion „**FindPosition(oElement)**“ verwendet. Damit kann mithilfe eines Mausklicks auf das Bild die Position des angeklickten Punktes innerhalb des Bildes bestimmt werden. Mit dieser Technik können verschiedene Bildausschnitte erfasst werden. Beispielsweise sei ein rechteckiger Bereich mit dem linken, oberen Punkt A(x1/y1) und der Breite a und der Höhe b vorgegeben. Der rechte, untere Diagonalen-Endpunkt B(x2/y2) hat die Koordianten x2 = x1+ a und y2 = y1 + b.
Der angeklickte Punkt sei P(x/y). Wenn nun P innerhalb dieses Bildbereichs liegt, dann soll ein bestimmter Befehl ausgeführt werden, was mit der nachfolgenden Codezeile erreicht wird.

if ((x > x1) && (x < x2) && (y > y1) && (y < y2)) { Befehl . . . }

Das nächsten zwei Programme „**imagepart.html**“ und „**imagegaps.html**“ realisieren diese Struktur.

(7.2.2) Auswahl von Image-Bereichen



Wird in diesem Programm der Bildausschnitt mit dem Foto von “John” angeklickt, dann wird in einem Meldungsfenster der Erfolg bestätigt, andernfalls erscheint eine Misserfolgsmeldung.

Programmlisting

```
<!DOCTYPE html>
<html>
<head>
<title>imagepart</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Image-Selection">
<meta name="author" content="Paukert Herbert">

<style>
body   { background-color: #CCCCEE; margin: 2%; padding-left:2%; font-size: 19px; }
#myImg { position: absolute; left: 420px; top: 80px; }
.ein   { margin: 2px; border: 1px solid #666666; background-color: #FFFFE8; font-size: 15px;}
.btn   { margin: 2px; border: 2px solid #666666; background-color: #EED8D8; font-size: 16px;
         border-radius: 4px;
}
</style>

<script>
var imgX, imgY;        // Koordinaten des angeklickten Bildpunktes
var normHeight = 450;  // normierte Bildhöhe für die Koordinaten-Abfrage
var result;            // Abfrage-Ergebnis

function setPosition(oElement,x,y) {
   let el = document.getElementById(oElement);
   el.style.position = "absolute";
   el.style.left = x + "px";
   el.style.top =  y + "px";
}

function setImage() {
   let imgHeight = document.getElementById("inp1").value;
   let imgLeft = document.getElementById("inp2").value;
   let imgTop = document.getElementById("inp3").value;
   document.getElementById("myImg").height = imgHeight;
   setPosition('myImg',imgLeft,imgTop);
}
```

```
function FindPosition(oElement) {
  if (typeof( oElement.offsetParent ) != "undefined") {
     for(var posX = 0, posY = 0; oElement; oElement = oElement.offsetParent) {
        posX += oElement.offsetLeft;
        posY += oElement.offsetTop;
     }
     return [ posX, posY ];
  }
  else { return [ oElement.x, oElement.y ]; }
}

function GetCoordinates(e,objEl) {
  let PosX = 0;
  let PosY = 0;
  let ImgPos = [0,0];
  ImgPos = FindPosition(objEl);
  if (!e) { var e = window.event; }
  if (e.pageX || e.pageY) {
     PosX = e.pageX;
     PosY = e.pageY;
  }
  else if (e.clientX || e.clientY) {
      PosX = e.clientX + document.body.scrollLeft
             + document.documentElement.scrollLeft;
      PosY = e.clientY + document.body.scrollTop
             + document.documentElement.scrollTop;
  }
  PosX = PosX - ImgPos[0];
  PosY = PosY - ImgPos[1];
  imgX = PosX;
  imgY = PosY;
}

function getImagePoint(event) {
   var myImg = document.getElementById("myImg");
   GetCoordinates(event,myImg);
   let h = myImg.height;
   let w = myImg.width;
   let k = w / h;            // aspect-ratio
   let h0 = normHeight;      // normierte Bildhöhe für die Koordinaten-Abfrage
   let w0 = h0 * k;
   imgX = Math.round(imgX * w0 / w);
   imgY = Math.round(imgY * h0 / h);
   //  alert(imgX + ' / ' + imgY);
   result = 0;
   // Rechteckiger Bildausschnitt von "John"
   x1 = 115; y1 = 230;
   x2 = 170; y2 = 310;
   if ( (imgX > x1) && (imgX < x2) && (imgY > y1) && (imgY < y2) ) {
      result ='1';
      alert('Das ist JOHN !');
   }
   else {
      result = '0';
      alert('Das ist NICHT John !');
   }
}

function refresh() {
  document.getElementById("inp1").value = "450";
  document.getElementById("inp2").value = "400";
  document.getElementById("inp3").value = "100";
  window.location.reload();
}
</script>
</head>

<body>
<h3>Select parts of an image (by mouseclick)</h3>
<span id="info">Click on the picture of "John"</span><br>
<br>
<input id="inp1" class="ein" type="text" value="450"> Image-Height<br>
<input id="inp2" class="ein" type="text" value="400"> Image-Left<br>
<input id="inp3" class="ein" type="text" value="100"> Image-Top<br>
<br>
<button id="btn1" class="btn" onclick = "setImage()">Image-Set</button><br>
<button id="btn2" class="btn" onclick = "refresh()">Page-Reset</button><br>
<br>
<img id="myImg" src='family.jpg' height='500' onclick='getImagePoint(event)'><br>
</body>
</html>
```

(7.2.3) Lückentext mit Image-Auswahl

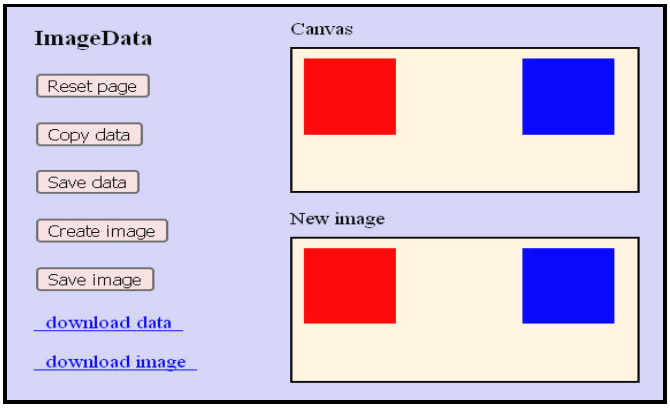
Programmlisting

```
<!DOCTYPE html>
<html>
<head>
<title>imagegaps</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="imagegaps">
<meta name="author" content="Paukert Herbert">

<style>
body   {background-color:#AAAADD; color:black; font-family:Calibri,Arial,sans-serif; font-size:19px;
        font-weight:normal; text-align:left; margin:2%; padding-left:2%;}
.cd1   {border: 1px solid #444444; background-color: #FFFFFF; font-size: 15px;}
.cd2   {border: 2px solid #444444; background-color: #EED8D8; font-size: 16px; font-weight: bold;
        border-radius: 6px;}
.p1    {color: blue; font-size: 16px; font-style: italic;}
.p2    {color: red; font-size: 19px; font-weight: bold;}
.list  {font-size: 19px;}
#msg   {color: blue; font-size: 18px; font-style: italic; font-weight: bold;}
#myImg {border: 2px solid #444444; border-radius: 16px; }
#fs    {width: 840px; background-color: #E8E8E8; border: 2px solid #444444; }
</style>

<script>
var imgX, imgY;          // Bildpunkt-Koordinaten
var normHeight = 450;    // normierte Bildhöhe für die Koordinaten-Abfrage
var aText = "";          // Antworttext

document.onclick = function(event){
  var ide = event.target.id;
  if ( (ide == "id0") && (window.getSelection()) ) {
    aText = myTrim(window.getSelection().toString());
  }
  else {
     if (ide != 'myImg') {
        if (aText != '') {  document.getElementById(ide).value = aText; aText = '';}
     }
  }
}

function myTrim(x)  { return x.replace(/^\s+|\s+$/gm,''); }
```

```
function FindPosition(oElement) {
  if (typeof( oElement.offsetParent ) != "undefined") {
     for(var posX = 0, posY = 0; oElement; oElement = oElement.offsetParent) {
        posX += oElement.offsetLeft;
        posY += oElement.offsetTop;
     }
     return [ posX, posY ];
  }
  else { return [ oElement.x, oElement.y ]; }
}

function GetCoordinates(e,objEl) {
  var PosX = 0;
  var PosY = 0;
  var ImgPos;
  ImgPos = FindPosition(objEl);
  if (!e) { var e = window.event; }
  if (e.pageX || e.pageY) {
     PosX = e.pageX;
     PosY = e.pageY;
  }
  else if (e.clientX || e.clientY) {
      PosX = e.clientX + document.body.scrollLeft
             + document.documentElement.scrollLeft;
      PosY = e.clientY + document.body.scrollTop
             + document.documentElement.scrollTop;
  }
  PosX = PosX - ImgPos[0]; imgX = PosX;
  PosY = PosY - ImgPos[1]; imgY = PosY;
}

function imgFunc(event) {
   var myImg = document.getElementById("myImg");
   GetCoordinates(event,myImg);
   if (myImg.height != normHeight) {
      let h = myImg.height;
      let w = myImg.width;
      let k = w / h;            // aspect-ratio
      let h0 = normHeight;      // normierte Bildhöhe für die Koordinaten-Abfrage
      let w0 = h0 * k;
      imgX = Math.round(imgX * w0 / w);
      imgY = Math.round(imgY * h0 / h);
   }
   aText = '0';
   // ------------------ Koordinaten mit “imageset.html” ermittelt ------------------
   x1=192; x2=205; x3=157; x4=170; x5=188; x6=206;
   x7=27; x8=49; x9=317; x10=334; x11=89; x12=112;
   x13=92; x14=118; x15=32; x16=48; x17=34; x18=56;
   x19=188; x20=211; x21=316; x22=335; x23=157; x24=181;
   y1=1; y2=18; y3=124; y4=140; y5=121; y6=142;
   y7=122; y8=141; y9=120; y10=138; y11=207; y12=222;
   y13=421; y14=439; y15=229; y16=247; y17=332; y18=351;
   y19=228; y20=251; y21=228; y22=251; y23=332; y24=355;

   if ( (imgX>x1) && (imgX<x2) && (imgY>y1) && (imgY<y2) ) {aText='1'; return;}
   if ( (imgX>x3) && (imgX<x4) && (imgY>y3) && (imgY<y4) ) {aText='2'; return;}
   if ( (imgX>x5) && (imgX<x6) && (imgY>y5) && (imgY<y6) ) {aText='3'; return;}
   if ( (imgX>x7) && (imgX<x8) && (imgY>y7) && (imgY<y8) ) {aText='4'; return;}
   if ( (imgX>x9) && (imgX<x10) && (imgY>y9) && (imgY<y10) ) {aText='5'; return;}
   if ( (imgX>x11) && (imgX<x12) && (imgY>y11) && (imgY<y12) ) {aText='6'; return;}
   if ( (imgX>x13) && (imgX<x14) && (imgY>y13) && (imgY<y14) ) {aText='7'; return;}
   if ( (imgX>x15) && (imgX<x16) && (imgY>y15) && (imgY<y16) ) {aText='8'; return;}
   if ( (imgX>x17) && (imgX<x18) && (imgY>y17) && (imgY<y18) ) {aText='9'; return;}
   if ( (imgX>x19) && (imgX<x20) && (imgY>y19) && (imgY<y20) ) {aText='10'; return;}
   if ( (imgX>x21) && (imgX<x22) && (imgY>y21) && (imgY<y22) ) {aText='11'; return;}
   if ( (imgX>x23) && (imgX<x24) && (imgY>y23) && (imgY<y24) ) {aText='12'; return;}
   // ------------------ Koordinaten mit “imageset.html” ermittelt ------------------
}

function getIndex(elem) {
  for (var i=0; i < document.myForm.elements.length; i++) {
      if (elem == document.myForm.elements[i]) { return i; }
  }
  return -1;
}

function showKey(ev) {
   var taste = ev.which || ev.keyCode;
   if (taste == 13)  {
      var field = ev.target;
      var i = getIndex(field);
      document.myForm.elements[i+1].focus();
   }
}
```

```
function pruef() {
  var formular = document.myForm;
  var anz = formular.length;
  var antwo = new Array(anz);
  antwo[0] = "4";
  antwo[1] = "2";
  antwo[2] = "6";
  antwo[3] = "1";
  antwo[4] = "3";
  antwo[5] = "5";
  antwo[6] = "11";
  antwo[7] = "10";
  antwo[8] = "8";
  antwo[9] = "7";
  antwo[10] = "12";
  antwo[11] = "9";
  for (var i = 0; i < anz; i++) { antwo[i] = myTrim(antwo[i]); }
  var num = 0;
  var plus = "+"
  for (var i = 0; i < anz; i++) {
    var s = antwo[i];
    if ( myTrim(formular.elements[i].value) == s ) {
       num++; formular.elements[i].value = s + plus;
    }
    else { formular.elements[i].value = s; };
  }
  alert(num + " von " + anz + " Treffern (+)");
  document.getElementById("msg").innerHTML = "(+) = richtige Antworten = " + num + " von " + anz;
}

function erase() {
  var formular = document.myForm;
  var anz = formular.length;
  for (var i = 0; i < anz; i++) { formular.elements[i].value = ""; }
  document.getElementById("msg").innerHTML = "(+) = richtige Antworten";
}
</script>
</head>

<body onkeydown = "showKey(event);" >
<fieldset id="fs">
<form name="myForm">
<table><tr>
<td width="50%">
<h3>My family (meine Familie)</h3>
Who is who? (Wer ist wer?)<br>My name is JOHN.<br>
<p class="p1">
Klicke zuerst auf eine Zahl im Bild<br>
und dann in die richtige Textlücke.<br>
Auch Tastatureingaben sind möglich.
</p>
<div class="list">
<input type="text" size="3" id="id1" class="cd1"/>     My father (mein Vater)<br>
<input type="text" size="3" id="id2" class="cd1"/>     My mother (meine Mutter)<br>
<input type="text" size="3" id="id3" class="cd1"/>     My parents (meine Eltern)<br>
<input type="text" size="3" id="id4" class="cd1"/>     My grandparents (meine Großeltern)<br>
<input type="text" size="3" id="id5" class="cd1"/>     My aunt (meine Tante)<br>
<input type="text" size="3" id="id6" class="cd1"/>     My uncle (mein Onkel)<br>
<input type="text" size="3" id="id7" class="cd1"/>     My sister (meine Schwester)<br>
<input type="text" size="3" id="id8" class="cd1"/>     My brother (mein Bruder)<br>
<input type="text" size="3" id="id9" class="cd1"/>     My wife (meine Frau)<br>
<input type="text" size="3" id="id10" class="cd1"/>    My children (meine Kinder)<br>
<input type="text" size="3" id="id11" class="cd1"/>    My son (mein Sohn)<br>
<input type="text" size="3" id="id12" class="cd1"/>    My daughter (meine Tochter)<br>
</div>
</td>
<td>
<br> <img id="myImg" src="family.jpg" height="450" alt="bild" onclick="imgFunc(event)">
</td>
</tr></table>
</form>
<br>
<input type="button" value=" Prüfen  " onclick="pruef()" class="cd2"/>  
<input type="button" value=" Löschen " onclick="erase()" class="cd2"/>  
<input type="button" value=" Drucken " onclick="print()" class="cd2"/>  
<br>
<span id = "msg">(+) = richtige Antworten</span><br>
<br>
<script> document.forms[0].elements[0].focus(); </script>
</fieldset>
</body>
</html>
```

(7.2.4) Manipulation von Image-Pixeln



Das Programm "**imagedata.html"** demonstriert einige wirkungsvolle Methoden des **ImageData**-Objekts. Mit diesen können die Bildpunkte (Pixel) eines Canvas-Objektes direkt angesprochen und kopiert werden. Wenn **context** der Canvas-Kontext ist, so wird mit **imgData = context.getImageData(x, y, width, height)** ein Canvas-Bereich auf der Variablen **imgData** gespeichert. Mit **putImageData(imgData, x1, y1)** wird dieser Bereich an die Offset-Koordinaten (x1/y1) wieder in die Canvasfläche kopiert. Dabei bleiben alle Farbwerte der Pixel erhalten. Zusätzlich besteht aber auch die Möglichkeit die Farbwerte der Pixel zu ändern, womit Grafikfilter erzeugt werden können. Dabei gibt es für jedes Pixel vier RGBA-Bytes, welche Werte von 0 bis 255 annehmen. Die Pixel sind in Zeilen angeordnet, die durch ein Array repräsentiert sind: **imgData.data[4 * (y * imgData.width + x) + n]** mit (x/y) als Koordinaten der Pixel in der Canvasfläche und mit n = 0 für Rot, n = 1 für Grün, n = 2 für Blau und n = 3 für den Alphawert (Opacitiy, Deckkraft). Die Formel des Array-Index dient der Transformation der zweidimensionalen Canvasfläche in das lineare Array.

Im Programm wird das Canvas-Objekt auf ein Image-Objekt abgebildet, wobei eine spezifische Daten-URL des Canvas-Objekts als Ressourcenzeiger auf die Canvasdaten verwendet wird. Die Objekte des vorliegenden Dokuments werden dynamisch erzeugt und zum Abschluss können die Bildinhalte von Canvas- und Image-Objekt mithilfe eingerichteter Links in den Download-Ordner abgespeichert werden.

Programmlisting

```
<!DOCTYPE html>
<html>
<head>
<title>imagedata</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Image-Data">
<meta name="author" content="Paukert Herbert">

<style>
body   { background-color: #CCCCEE; margin: 2%; font-size: 19px; }
.btn   { margin: 2px; border: 2px solid #666666; background-color: #EED8D8;
         font-size: 16px; border-radius: 4px; }
#info1 { position: absolute; left: 250px; top: 20px; }
#info2 { position: absolute; left: 250px; top: 220px; visibility: hidden; }
</style>

<script>
var myCanvas;  // Canvas-Variable
var context;   // Canvas-Context
var imgData;   // Image-Daten auf dem Canvas
var myImage;   // Image-Variable
var okay;      // Steuervariable

function init() {
// Dynamische Erzeugung eines Canvas-Objektes
  myCanvas = document.createElement('canvas');
  myCanvas.style.position = 'absolute';
  myCanvas.style.left = '250px';
  myCanvas.style.top  = '50px';
  myCanvas.style.clientwidth  = '500px';
  myCanvas.style.clientheight = '200px';
  myCanvas.style.border = '2px solid black';
```

```
  document.body.appendChild(myCanvas);
  context = myCanvas.getContext("2d");
  context.fillStyle = "antiquewhite";
  context.fillRect(0, 0, myCanvas.width, myCanvas.height);
  context.fillStyle = "red";
  context.fillRect(10, 10, 80, 80);
}

function change() {
// Verändert die Farbwerte der Bildpunkte (Pixelmanipulation)
  for (var y = 0; y < imgData.height; y++) {
     for (var x = 0; x < imgData.width; x++) {
        imgData.data[4 * (y * imgData.width + x) + 0] = 0;    // Rotwert
        imgData.data[4 * (y * imgData.width + x) + 1] = 0;    // Grünwert
        imgData.data[4 * (y * imgData.width + x) + 2] = 255;  // Blauwert
        imgData.data[4 * (y * imgData.width + x) + 3] = 255;  // Alphawert (Deckkraft)
     }
  }
}

function copy() {
// Daten kopieren innerhalb des Canvas-Objektes
  imgData = context.getImageData(10, 10, 80, 80);   // Canvasdaten zwischenspeichern
  change();                                         // zwischengespeicherte Daten ändern
  context.putImageData(imgData, 200, 10);           // geänderte Daten zurückspeichern
}

function store() {
// Canvas-Daten als Bild speichern
  anchor = document.createElement("a");
  anchor.href = myCanvas.toDataURL("image/png");
  anchor.download  = "canvas.png";
  anchor.innerHTML = "  download data  ";
  document.body.appendChild(anchor);
  p = document.createElement('p');
  document.body.appendChild(p);
  anchor.click;
}

function create() {
// Dynamische Erzeugung eines Image-Objektes
  dataUrl = myCanvas.toDataURL();
  myImage = document.createElement('img');
  myImage.src = dataUrl;
  myImage.style.position = 'absolute';
  myImage.style.left   = '250px';
  myImage.style.top    = '250px';
  myImage.style.width  = myCanvas.width;
  myImage.style.height = myCanvas.height;
  myImage.style.border = '2px solid black';
  document.body.appendChild(myImage);
  document.getElementById("info2").style.visibility = "visible";
  okay = true;
}

function save() {
// Image-Daten als Bild speichern
  if (!okay) { alert('Zuerst [Create] ausführen!'); return; }
  link = document.createElement('a');
  link.href = myImage.src;
  link.download = "image.jpg";
  link.innerHTML = "  download image  ";
  document.body.appendChild(link);
  p = document.createElement('p');
  document.body.appendChild(p);
  link.click();
}

function reset() {  window.location.reload(); }

</script>
</head>

<body onload = "init()">
<h3>ImageData</h3>
<button class="btn" onclick="reset()">Reset page</button><br><br>
<button class="btn" onclick="copy()">Copy data</button><br><br>
<button class="btn" onclick="store()">Save data</button><br><br>
<button class="btn" onclick="create()">Create image</button><br><br>
<button class="btn" onclick="save()">Save image</button><br><br>
<span id="info1">Canvas</span>
<span id="info2">New image</span>
</body>
</html>
```

(7.2.5) Bildverarbeitung und Image-Filter

Das Programm "**imagepix.html**" ist eine einfache Bildverarbeitung (Upload, Download, Zoom, Rotation und Crop) und verwendet außerdem verschiedene Image-Filter um die Farbmerkmale eines Bildes zu verändern.

Die Grafik zeigt oben das Auswahl-Menü.

Links ist ein Bild in originaler Größe.

In der Mitte ist das Bild um 50% verkleinert und dann zusätzlich um 90° gedreht.

Auf der rechten Seite ist unten ein Teil des Bildes ausgeschnitten (Crop) und dann vergrößert.

Der Befehl `ctx.drawImage(myImage,0,0,myImage.width,myImage.height);` bildet das Bild „myImage" in ganzer Größe auf den Canvas ab, mit Startpunkt (0,0) und mit ganzer Bildbreite und Bildhöhe.

Der Befehl `myImage.src = myCanvas.toDataURL("image/jpeg",0.90);` bildet den Canvas-Inhalt auf das Bild „myImage" ab, mithilfe der Daten-URL des Canvas-Objektes und mit 90% Bildgenauigkeit.

Die anderen Befehle im Listing wurden in den vorangehenden Buchkapiteln bereits ausführlich erklärt.

Programmlisting

```
<!DOCTYPE html>
<html>
<head>
<title>imagepix.html</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Bildbearbeitung, Bildfilter">
<meta name="author" content="Paukert Herbert">

<style>
body { background-color: #CCCCEE; margin: 20px; font-family:Arial; font-size: 16px; }
#pic { z-index: 2; }
#canvas { z-Index: 1; }
#fs  { width: 540px; border: 2px solid #666666; background-color: #DDDDDD;  z-index: -1; }
.btn { margin: 4px; border: 2px solid #666666; background-color: #EED8D8;
       font-size: 16px; border-radius: 4px; }
.inp { margin: 2px; border: 1px solid #666666; background-color: white;
       font-size: 16px; border-radius: 4px; }
#fileInput { font-size: 16px; }
#downld { color: red; font-style: italic; font-weight: bold; }
</style>

<script>
var sTime = 100;   // Wartezeit für Sleep-Funktion (Msec)

function Sleep(msec) {
// Bewirkt in asynchronen Funktionen ein Warten
   return new Promise(resolve => setTimeout(resolve,msec));
}

var myImage  = new Image();           // Bild
var myImage0 = new Image();           // Originalbild

var myCanvas;                         // Canvas-Objekt-Variable
var ctx;                              // Canvas-Context-Variable
var imgdata;                          // Image-Daten auf dem Canvas
```

```
var fZoom = 50;                       // Zoomfaktor (%)
var rStep = 90;                       // Rotationsschrittweite (°)
var rWin = 0;                         // Rotationswinkel
var fTyp = 0;                         // Filtertyp
var fInt = 1;                         // Filterintensität
var fStep = 0.10;                     // Filterschrittweite
var fDir = 1;                         // Filterrichtung
var original = false;                 // Kennvariable für Originalbild
var newOrg = false;                   // Kennvariable für Original = Zuschnitt (crop)
var rotated = false;                  // Kennvariable für Rotation
var zoomed = false;                   // Kennvariable für Zoom
var cropped = false;                  // Kennvariable für Zuschnitt
var x0,y0,z0,x,y,z;                   // Hilfsvariable für "width" und "height"
var xa,ya,xe,ye,dx,dy;                // Markierungsrahmen
var marked = false;                   // Kennvariable für Markierung

function increase() { fDir = (+1); }  // Filterintensität zunehmend
function decrease() { fDir = (-1); }  // Filterintensität abnehmend

function initCanvas() {
// Canvas initialisieren
  myCanvas = document.getElementById('canvas');
  myCanvas.style.cursor = "crosshair";
  myCanvas.style.filter = "none";
  ctx = myCanvas.getContext("2d");
}

function myFilter() {
// Filtern - Teil 1
  fInt = fInt + fDir*fStep;
  if (fTyp == 4) { ctx.filter = "brightness(" + fInt + ")"; }
  if (fTyp == 5) { ctx.filter = "contrast(" + fInt + ")"; }
  if (fTyp == 6) { ctx.filter = "saturate(" + fInt + ")"; }
}

function filterImage(n) {
// Filtern - Teil 2
   fTyp = n;
   if (fTyp == 1) { ctx.filter = "grayscale(1)"; }
   if (fTyp == 2) { ctx.filter = "sepia(1)"; }
   if (fTyp == 3) { ctx.filter = "invert(1)"; }
   if ((fTyp == 4) || (fTyp == 5) || (fTyp == 6)) { myFilter(); }
   if (zoomed && !cropped) {
       myImage.width = parseInt(x/fZoom);
       myImage.height = parseInt(y/fZoom);
   }
   ctx.drawImage(myImage,0,0,myImage.width,myImage.height);
}

function zoom() {
// Bild fortgesetzt zoomen
   ctx.clearRect(0, 0, canvas.width, canvas.height);
   let s = document.getElementById('ein1').value;
   fZoom = 1*s/100;
   x = parseInt(fZoom * myImage.width);
   y = parseInt(fZoom * myImage.height);
   canvas.width = x; canvas.height = y;
   if (rotated) {
     z = x;
     if (y > x) { z = y; }
     canvas.width = z; canvas.height = z;
   }
   ctx.scale(fZoom,fZoom);
   ctx.drawImage(myImage,0,0,myImage.width,myImage.height);
   myImage.width = x; myImage.height = y;
   zoomed = true;
   original = false;
}

function rot() {
// Bild fortgesetzt rotieren um 90°
   if (zoomed) { rWin = 0; zoomed = false; }
   rWin = rWin + rStep;
   if (rWin >= 360) { rWin = 0; }
   myImage.width = x; myImage.height = y;
   canvas.width = x; canvas.height = y;
   z = x;
   if (y > x)  { z = y; }
   canvas.width = z; canvas.height = z;
   ctx.clearRect(0, 0, z, z);
   ctx.translate(z/2,z/2);
   ctx.rotate(rWin*Math.PI/180);
   ctx.translate(-z/2,-z/2);
```

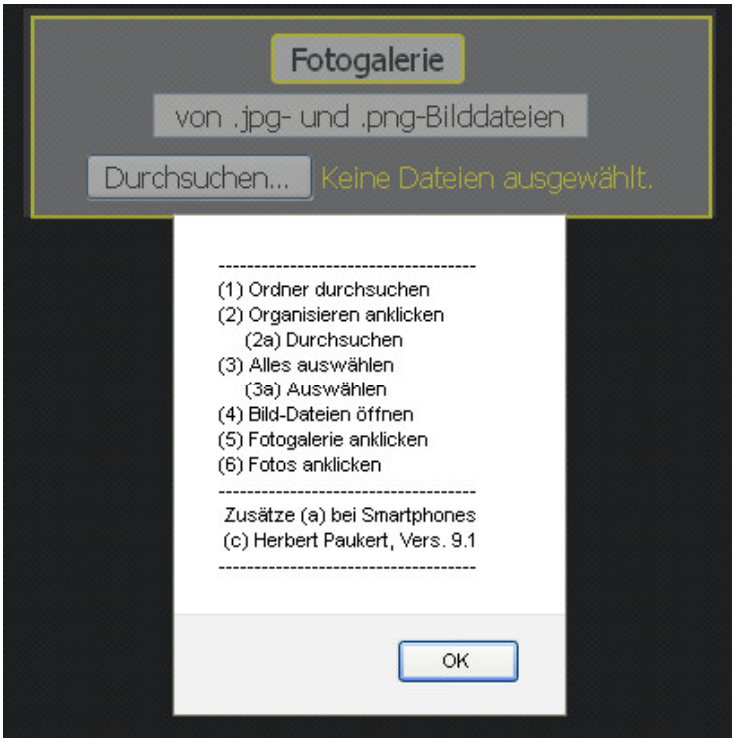
```
   if (rWin == 90 && y > x)  { }
   if (rWin == 180 && y > x) { ctx.translate(z-x,0); }
   if (rWin == 270 && y > x) { ctx.translate(z-x,0); }
   if (rWin == 0 && y > x)   {  }

   if (rWin == 90 && y < x)  { ctx.translate(0,z-y); }
   if (rWin == 180 && y < x) { ctx.translate(0,z-y); }
   if (rWin == 270 && y < x) {  }
   if (rWin == 0 && y < x)   {  }

   ctx.drawImage(myImage,0,0,x,y);
   rotated = true;
   original = false;
}

function crop90() {
// Bild zuschneiden bei Drehwinkel von (n * 90°)
   if (rWin == 90 && y > x)  {
      imgdata = ctx.getImageData(0,0,z,x);
      ctx.clearRect(0, 0, z, z);
      canvas.width = z; canvas.height = x;
   }
   if (rWin == 180 && y > x)  {
      imgdata = ctx.getImageData(0,0,x,z);
      ctx.clearRect(0, 0, z, z);
      canvas.width = x; canvas.height = z;
   }
   if (rWin == 270 && y > x)  {
      imgdata = ctx.getImageData(0,0,z,x);
      ctx.clearRect(0, 0, z, z);
      canvas.width = z; canvas.height = x;
   }
   if (rWin == 0 && y > x)  {
      imgdata = ctx.getImageData(0,0,x,z);
      ctx.clearRect(0, 0, z, z);
      canvas.width = x; canvas.height = z;
   }

   if (rWin == 90 && y < x)  {
      imgdata = ctx.getImageData(0,0,x,z);
      ctx.clearRect(0, 0, z, z);
      canvas.width = y; canvas.height = z;
   }
   if (rWin == 180 && y < x)  {
      imgdata = ctx.getImageData(0,0,z,x);
      ctx.clearRect(0, 0, z, z);
      canvas.width = x; canvas.height = y;
   }
   if (rWin == 270 && y < x)  {
      imgdata = ctx.getImageData(0,0,x,z);
      ctx.clearRect(0, 0, z, z);
      canvas.width = y; canvas.height = z;
   }
   if (rWin == 0 && y < x)  {
      imgdata = ctx.getImageData(0,0,z,x);
      ctx.clearRect(0, 0, z, z);
      canvas.width = x; canvas.height = y;
   }
   ctx.putImageData(imgdata,0,0);
}
function cropShow() {
// Bild zuschneiden
   if (!cropped) { alert('Abbruch: Noch kein Bildzuschnitt mit zwei Maus-Klicks!'); return; }
   cropped = false;
   imgdata = ctx.getImageData(xa,ya,dx,dy);
   ctx.clearRect(0, 0, x0, y0);
   canvas.width = dx; canvas.height = dy;
   ctx.putImageData(imgdata,0,0);
   myImage.width = dx; myImage.height = dy;
   myImage.src = myCanvas.toDataURL("image/jpeg",0.90);
   x = dx;
   y = dy;
   newOrg = true;
}
function crop(ev) {
// Eventhandler für "mousedown" zum Ausschneiden von Bildteilen mit zwei Mausklicks
// (erster Mausklick auf links-oben, zweiter Mausklick auf rechts-unten des Bildteils)
  if (!original) { alert('Abbruch: Bildzuschnitt nur vom Original möglich!'); return; }
  cropped = true;
  var ide = "canvas";
  var canvas = document.getElementById("canvas");
  var ctx = canvas.getContext("2d");
  var cData = document.getElementById(ide).getBoundingClientRect();
```

```
  if (marked) {
    xe = parseInt(ev.clientX - cData.left);
    ye = parseInt(ev.clientY - cData.top);
    if ((xe < xa) || (ye < ya)) {
       alert('Abbruch: Falsche Markierung !')
       marked = false;
       return;
    }
    dx = xe - xa;
    dy = ye - ya;
    ctx.beginPath();
    ctx.lineWidth = 2;
    ctx.strokeStyle = 'white';
    ctx.strokeRect(xa,ya,dx,dy);
    ctx.stroke();
    // alert('Bildpunkt: ' + xe + ' /  ' + ye);
    marked = false;
    alert('Weiter mit [crop]');
    // window.scrollTo(0,0);
    return;
  }
  if (!marked) {
    xa = parseInt(ev.clientX - cData.left);
    ya = parseInt(ev.clientY - cData.top);
    // alert(ide + ': ' + cData.width + ' /  ' + cData.height + ' // Position: ' + xa + ' / ' + ya);
    marked = true;
  }
}
function Restore() {
// Originales Bild aufrufen
   if (newOrg) {
      okay = confirm('Soll der Zuschnitt zum Original werden?');
      if (okay) {
        x0 = dx;
        y0 = dy;
        myImage0.width = dx; myImage0.height = dy;
        myImage0.src = myImage.src;
      }
      newOrg = false;
   }
   original = true;
   rotated = false;
   zoomed = false;
   fDir = 1;
   fInt = 1;
   fStep = 0.10;
   rWin = 0;
   rStep = 90;
   fZoom = 50;
   document.getElementById('ein1').value = fZoom;
   ctx.filter = "none";
   x = x0;
   y = y0;
   canvas.width = x0; canvas.height = y0;
   myImage.width = x0; myImage.height = y0;
   myImage.src = myImage0.src;
   ctx.drawImage(myImage,0,0,myImage.width,myImage.height);
   alert('Originale Bildabmessungen: ' + myImage.width + ' x ' + myImage.height);
}
async function loadFile() {
// Eine Grafikdatei angepasst in den Canvas laden
   ctx.clearRect(0, 0, canvas.width, canvas.height);
   var fileToLoad = document.getElementById('fileInput').files[0];
   var reader = new FileReader();
   imageType = /image./;
   if ( !fileToLoad.type.match(imageType) ) {
        document.getElementById('fileInput').value = '';
        alert('Falscher Dateityp');
        return;
   }
   reader.readAsDataURL(fileToLoad);
   reader.addEventListener("load", function() {
       myImage.src = reader.result;
   }, false);
   myImage.onload = function() {
       canvas.width = myImage.width; canvas.height = myImage.height;
       ctx.drawImage(myImage,0,0,myImage.width,myImage.height);
   }
   var name = fileToLoad.name;
   await Sleep(sTime);  // async
```

```
   alert('Bild [' + name + '] erfolgreich geladen!');
   document.getElementById("ein").value = name;
   x0 = myImage.width; y0 = myImage.height;
   myImage0.width = x0; myImage0.height = y0;
   myImage0.src = myImage.src;
   Restore();
}
function saveFile() {
// Den Canvas in eine Grafikdatei speichern
   if (rotated) { crop90(); }
   var name = document.getElementById('ein').value;
   name = name.trim();
   if (name == "") { alert('Dateiname fehlt !'); return; }
   link = document.getElementById('downld');
   link.href = myCanvas.toDataURL("image/jpeg",0.90);
   link.download = name;
   alert('Bild [' + name + '] im "Download-Ordner" gespeichert!');
}
function printGraph() {
// Grafik drucken
   document.getElementById("fs").style.visibility = 'hidden';
   document.getElementById("pic").style.top = "0px";
   window.print();
   document.getElementById("pic").style.top = "220px";
   document.getElementById("fs").style.visibility = 'visible';
}
function Reset() {
// HTML-Datei neu laden
   document.getElementById('fileInput').value = '';
   document.getElementById('ein').value = '';
   location.reload();
}
function Info() {
// Kurzinformation
  inf = 'Reihenfolge der Bild-Verarbeitung:' + '\n' +
        'Upload-Zoom-Rotate-Filter-Download' + '\n' +
        'Stärker(+) oder schwächer(-) Filtern.' + '\n' +
        'Vor jedem [Upload] immer [Reset].' + '\n' +
        'Zuschneiden eines Bild-Bereiches:' + '\n' +
        'Zuerst [Originales Bild] aufrufen,' + '\n' +
        'dann erster Mausklick auf links oben' + '\n' +
        'und zweiter Klick auf rechts unten' + '\n' +
        'des gewünschten Bild-Bereiches.' + '\n' +
        'Zuletzt [Crop] aufrufen.'  + '\n' +
        'Dieser zugeschnittene Bild-Bereich ' + '\n' +
        'kann dann weiter bearbeitet werden.' + '\n' +
        '(c) Herbert Paukert, Vers. 4.0, 2023';
   alert(inf);
}
</script>
</head>
<body onload="initCanvas()">
<div id="pic" style="position:absolute; left:20px; top:220px;">
  <canvas id="canvas" width="640" height="480" onclick="crop(event)">
</div>
<fieldset id="fs">
<button type="button" class="btn" onclick="Info()">Info</button>
<button type="button" class="btn" onclick="Restore()">Originales Bild</button>
<input type="text" name="ein1" id="ein1" class="inp" value="50" size="2">%
<button type="button" class="btn" onclick="zoom()">Zoom</button>
<button type="button" class="btn" onclick="rot()">Rotate (90°)</button>
<button type="button" class="btn" onclick="cropShow()">Crop</button><br>
<button type="button" class="btn" onclick="filterImage(1)">Grau</button>
<button type="button" class="btn" onclick="filterImage(2)">Sepia</button>
<button type="button" class="btn" onclick="filterImage(3)">Inversion</button>
<button type="button" class="btn" onclick="filterImage(4)">Helligkeit</button>
<button type="button" class="btn" onclick="filterImage(5)">Kontrast</button>
<button type="button" class="btn" onclick="filterImage(6)">Sättigung</button><br>
<button type="button" class="btn" onclick="increase()">Filter (+)</button>
<button type="button" class="btn" onclick="decrease()">Filter (-)</button>
<button type="button" class="btn" onclick="printGraph()">Print</button>
<button type="button" class="btn" onclick="Reset()">Reset</button><br>
Image upload: 
<input type="file" id="fileInput" accept=".jpg,.png" onchange="loadFile()"><br>
Dateiname: <input type="text" id="ein" class="inp" value=" " size="12">
<u><a id="downld" onclick="saveFile()">Image download</a></u>
</fieldset>
</body>
</html>
```

(7.2.6) Eine Image-Galerie

Das Programm „**paufoto.html**“ erzeugt eine Galerie von beliebig vielen Bildern aus einem ausgewählten Ordner des Computers, d.h. es werden Miniaturgrafiken (Thumbnails) der Bilder in einer Tabelle aufgelistet. Der Zugriff auf die Bilddateien eines Ordners erfolgt genau so wie in „**pauview.html**“. Die zwei folgenden Abbildung zeigen den Startbildschirm des Programms und eine Fotogalerie. Durch Anklicken eines Bildes kann das Bild vergrößert dargestellt werden. Erklärungen zum Programm findet man am Ende des Programmlistings.




```
<!DOCTYPE html>
<head>
<title>Fotogalerie</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Fotogalerie aus einem beliebigen Ordner">
<meta name="author" content="Herbert Paukert">
```

```
<style type="text/css">
   #bo    { background-color:#000; color:yellow; font-family:sans-serif; font-size:10px;
            font-weight:normal;
            text-align:center; margin-left: 10px; overflow: visible; }
   #menu  { background-color: #666; width: 360px; border: 2px solid yellow; margin: auto;
            padding: 8px; font-size: 16px;}
   .btn   { background-color: #DDD; border: 2px solid yellow; border-radius:4px; font-size: 18px;}
   #input { font-size: 16px; margin-top: 8px; }
   #fname { font-size: 16px; margin-top: 5px; text-align:center; }
   #ta    { margin:auto; width:1000px; table-layout: fixed; }
</style>

<script>
   // paufoto.html, Version 9.1
   var info = "-----------------------------------\n" +
              "(1) Ordner durchsuchen\n" +
              "(2) Organisieren anklicken\n" +
              "     (2a) Durchsuchen\n" +
              "(3) Alles auswählen\n" +
              "     (3a) Auswählen\n" +
              "(4) Bild-Dateien öffnen\n" +
              "(5) Fotogalerie anklicken\n" +
              "(6) Fotos anklicken\n" +
              "-----------------------------------\n" +
              " Zusätze (a) bei Smartphones        \n" +
              " (c) Herbert Paukert, Vers. 9.1     \n" +
              "-----------------------------------\n";

   var okay = false;                    // Steuervariable
   var pictA = new Image();             // Bild-Speicher
   var flist, file, fmax;               // File-Variable
   var num = 0;                         // File-Zähler
   var vbody, vtable, vtr, vtd;         // Objekt-Variable der Bildergalerie
   var anz = 10;                        // Anzahl der Thumbnails
   var Fenster;                         // Fenster-Objekt
   var miniWidth = 80;                  // Thumbnail-Breite
   var miniHeight = 80;                 // Thumbnail-Höhe
   var smallScr = false;                // Steuervariable für Smartphones

   window.addEventListener( "unload", function() {Fenster.close();} );

   function PopUp(pict){
   // PopUp-Fenster öffnen
     if (Fenster) { Fenster.close(); }
     Fenster = window.open("","","top=20,left=20,width=900 height=700,resizable=yes,scrollbars=yes");
     var b = "<IMG SRC = " + pict + " height=90%>";
     with (Fenster.document) {
        writeln("<HTML><HEAD><TITLE></TITLE></HEAD>");
        writeln("<BODY style='background-color: #333; text-align: center'>");
        writeln("<br>");
        writeln(b);
        writeln("<br>");
        writeln("</BODY></HTML>");
     }
   }

   document.addEventListener("DOMContentLoaded", function() {
   // Anfangsinformation
      if (screen.width < 800) { smallScr = true; }
      else  { smallScr = false; }
      var tabl = document.getElementById("ta");
      if (smallScr) {
        anz = 4;
        miniWidth = 70;
        miniHeight = 70;
        tabl.style.width = screen.width +"px";
        tabl.style.margin = "0";
        document.getElementById("bo").style.margin = "0";
        document.getElementById("bo").style.fontSize = "6";
        document.getElementById("menu").style.margin = "0";
      }
      alert(info);
   });

   function selectFile() {
   // Bilddateien auswählen
     okay = true;
     var fileInput = document.getElementById("input");
     flist = fileInput.files;
     fmax = flist.length;
   }
```

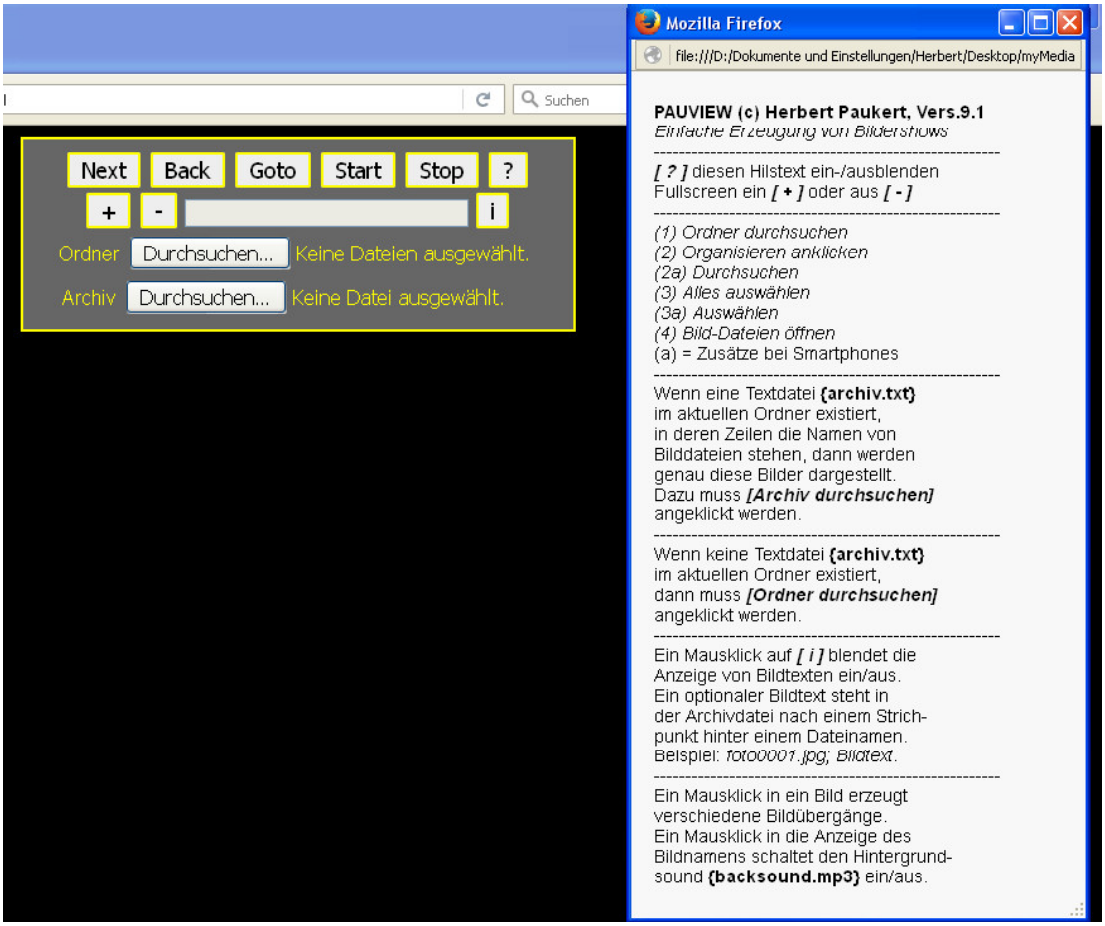
```
   function readFile(data,num) {
   // Bilddatei einlesen und im PopUp-Fenster darstellen
     var j = num + 1;
     var reader = new FileReader();
     reader.readAsArrayBuffer(data);
     reader.onload = function() {
        var buffer = reader.result;
        var blob = new Blob([new Uint8Array(buffer)]);
        var url = window.URL.createObjectURL(blob);
        pictA.src = url;
        var vimg = document.createElement("IMG");
        vimg.setAttribute("src",pictA.src);
        vimg.setAttribute("id","img" + j);
        vimg.setAttribute("width",miniWidth);
//        vimg.setAttribute("height",miniHeight);
        vimg.setAttribute("alt","no image");
        vimg.addEventListener("click", function() { PopUp(this.src); });
        var tcell = document.getElementById("cell"+ j);
        if (!smallScr) {
           var fname = document.createTextNode(data.name + '\n');
           tcell.appendChild(fname);
        }
        tcell.appendChild(vimg);
        document.getElementById("img" + j).style.border='2px solid silver';
        document.getElementById("img" + j).style.margin='4px';
     }
   }

   function Galeria() {
   // Bildergalerie erzeugen
      if (!okay) { return; }
      vbody = document.getElementById("bo");
      vtable = document.getElementById("ta");
      for (var num = 1; num <= fmax; num++) {
        if ( (num % anz) == 1 ) {
           vtr = document.createElement("TR");
           vtable.appendChild(vtr);
        }
        vtd = document.createElement("TD");
        vtd.setAttribute("id","cell" + num);
        vtr.appendChild(vtd);
      }
      for (var num = 0; num < fmax; num++) {
        file = flist[num];
        readFile(file,num);
      }
   }
</script>
</head>

<body id="bo">
<div id="menu">
  <input type = "button" class = "btn" value = " Fotogalerie " onclick = "Galeria()"> 
  <br>
  <input type = "text" id="fname" value=" von .jpg- und .png-Bilddateien " size="30" readonly>
  <br>
  <input type="file" multiple id="input" accept=".jpg,.png" onchange = "selectFile()">
  <br>
</div>
<br>
  <table id = "ta"> </table>
  <br>
</body>
</html>
```

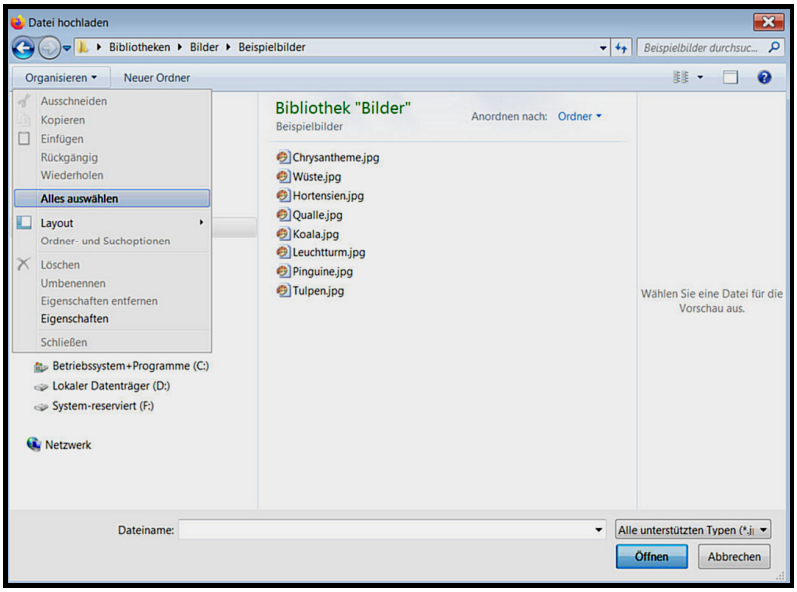
Der HTML-Tag <**input type =“file“ . . .**> ermöglicht JavaScript einen lesenden Zugriff auf Dateien und auf einschlägige Datei-Informationen wie Name, Größe, Typ. Das Attribut „**multiple**“ erlaubt die Auswahl und Eingabe von mehreren Werten. Mithilfe des Attributes „**files**“ wird eine Dateiliste „**flist**“ als Array erstellt. Das erfolgt in der Funktion „**selectFile()**“, die als „**onchange**“-Eventhandler dem HTML-Tag angehängt ist. In einem Fenster können ein Ordner des Computers und die Dateien ausgewählt werden. Deren Namen sind dann als nummerierte Items im „**flist**“-Array abgespeichert. Die Anzahl dieser Dateien liefert die Methode „**flist.length**“.

In der Funktion „**Galeria()**“ wird eine Tabelle von Thumbnails aller erfassten Bilder erzeugt. Dabei wird die Funktion „**readFile(file,num)**“ verwendet, welche das Bild mit dem Index „**num“** des Bilderarrays „**flist“** aus dem Ordner in die Tabelle lädt.

(7.2.7) Eine Image-Show

Das Programm „**pauview.html**“ermöglicht die animierte Darstellung von beliebig vielen Bildern aus einem ausgewählten Ordner des Computers. Die zwei folgenden Abbildungen zeigen den Startbildschirm mit dem Hilfefenster des Programms und zusätzlich ein Ordnerfenster mit den dort gespeicherten Bilddateien. Erklärungen zum Programm findet man am Ende des Programmlistings.





Hinweis: In JavaScript ist es **nicht** möglich von eingelesenen Dateien den lokalen Pfad zu extrahieren!

Programmlisting

```
<!DOCTYPE html>
<head>
<title>Fotoshow</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Fotoshow aus einem beliebigen Ordner">
<meta name="author" content="Herbert Paukert">

<style type="text/css">
   body  { background-color:#000; color:yellow;
           font-family:sans-serif; font-size:16px; font-weight:normal;
           text-align:center; margin-top: 10px; overflow: hidden; }
   #info { background-color: gray; width: 380px; height: 20px; border: 1px solid silver; padding: 2px;
           position: absolute; left: 10px; top:10px; visibility: hidden; color: yellow; zIndex=10; }
   #menu { background-color: #666; width: 420px; border: 2px solid yellow; margin:auto; padding: 8px; }
   #input  { font-size:16px; margin: 4px; }
   #input1 { font-size:16px; margin: 4px; }
   #pict   { border:0px solid gray; }
   #pict1  { border:0px solid gray; }
   #fname  { margin-top: 5px; }
   .btn    { background-color: #EEE; border: 2px solid yellow; border-radius:2px; font-size: 18px;
             margin: 2px; }
</style>

<script>
   // pauview.html, Version 9.1
   var iText = " \n" +
               "-----------------------------------------\n" +
               "  PAUVIEW, Bildershows erzeugen\n" +
               "  (c) Herbert Paukert, Version 9.1\n" +
               "  Programmanleitung mit [ ? ] lesen\n" +
               "-----------------------------------------\n";

   var okay = false;           // allgemeine Steuervariable
   var pictA = new Image();    // Bild-Speicher
   var pictB = new Image();    // Bild-Speicher
   var bild;                   // Image-Variable
   var blenden = 9;            // Anzahl der Bildübergänge
   var animat = 0;             // Kennzahl der Bildübergänge
   var antime = 2000;          // Dauer der Bildübergänge
   var zeit  = 5000;           // Bilder-Darbietungszeit für Show
   var ShowRun = false;        // Steuervariable für Show
   var active;                 // Steuervariable für TimeOut bei Shows
   var url;                    // Adressenzeiger einer Bilddatei
   var datNames;               // Stringvariable der Bilddatei-Namen
   var flist = [];             // Array der Bilddatei-Namen
   var file;                   // Ein Listenelement
   var max;                    // Anzahl der Listenelemente
   var num;                    // Aktuelle Nummer eines Listenelementes
   var even = true;            // Geradzahlige Nummer

   var start = false;          // Steuervariable für Bilderarchiv
   var sound = false;          // Toggle-Variable für Hintergrundsound

   var fadeTime = 1000;        // Dauer der Fading-Blende (MSec)
   var fadeTime0 = 1000;       // Standard-Blendendauer
   var fadeRun = false;        // Toggle für Fading-Blende

   var info = false;           // Toggle für Informationsfenster
   var alist = [];             // Hilfsarray
   var blist = [];             // Array der Bildinformationen

   var fullScr = false;        // Steuervariable für Fullscreen
   var smallScr = false;       // Steuervariable für Smallscreen von Smartphones

   var childWindow;
   var child = false;
   window.addEventListener( "unload", function() { closeChild(); } );

   function closeChild() {
   // Hilfetext schließen
     if (!childWindow || childWindow.closed) { return; }
     if (child) { childWindow.document.close(); childWindow.close(); }
   }
```

```
function showHelp() {
// Hilfetext anzeigen
  if (child) { closeChild(); child = false; return; }
  child = true;
  childWindow = window.open('','childWindow','top=20, left=1000, width=360, height=720,
       srollbars=yes, menubar=no, toolbar=no');
  childWindow.document.open();
  childWindow.document.write('<html><head><meta name="viewport" content="width=device-width,
       initial-scale=1.0"></head>');
  childWindow.document.write('<body style="background-color:#F8F8F8; font-family: sans-serif;
       font-size:14px; text-size-adjust:none; padding-left:10px; padding-bottom:10px;">');
  childWindow.document.write('<br><b>PAUVIEW (c) Herbert Paukert, Vers.9.1</b><br>');
  childWindow.document.write('<i>Einfache Erzeugung von Bildershows</i><br>');
  childWindow.document.write('-----------------------------------------------------------<br>');
  childWindow.document.write('<b><i>[ ? ]</i></b> diesen Hilstext ein-/ausblenden<br>');
  childWindow.document.write('Fullscreen ein <b><i>[ + ]</i></b> oder aus <b><i>
       [ - ]</i></b><br>');
  childWindow.document.write('-----------------------------------------------------------<br>');
  childWindow.document.write('<i>(1) Ordner durchsuchen<br>');
  childWindow.document.write('(2) Organisieren anklicken<br>');
  childWindow.document.write('     (2a) Durchsuchen<br>');
  childWindow.document.write('(3) Alles auswählen<br>');
  childWindow.document.write('     (3a) Auswählen<br>');
  childWindow.document.write('(4) Bild-Dateien öffnen</i><br>');
  childWindow.document.write('(a) = Zusätze bei Smartphones<br>');
  childWindow.document.write('-----------------------------------------------------------<br>');
  childWindow.document.write('Wenn eine Textdatei <b>{archiv.txt}</b><br>');
  childWindow.document.write('im aktuellen Ordner existiert,<br>');
  childWindow.document.write('in deren Zeilen die Namen von<br>');
  childWindow.document.write('Bilddateien stehen, dann werden<br>');
  childWindow.document.write('genau diese Bilder dargestellt.<br>');
  childWindow.document.write('Dazu muss <b><i>[Archiv durchsuchen]</i></b><br>');
  childWindow.document.write('angeklickt werden.<br>');
  childWindow.document.write('-----------------------------------------------------------<br>');
  childWindow.document.write('Wenn keine Textdatei <b>{archiv.txt}</b><br>');
  childWindow.document.write('im aktuellen Ordner existiert,<br>');
  childWindow.document.write('dann muss <b><i>[Ordner durchsuchen]</i></b><br>');
  childWindow.document.write('angeklickt werden.<br>');
  childWindow.document.write('-----------------------------------------------------------<br>');
  childWindow.document.write('Ein Mausklick auf <b><i>[ i ]</i></b> blendet die<br>');
  childWindow.document.write('Anzeige von Bildtexten ein/aus.<br>');
  childWindow.document.write('Ein optionaler Bildtext steht in<br>');
  childWindow.document.write('der Archivdatei nach einem Strich-<br>');
  childWindow.document.write('punkt hinter einem Dateinamen.<br>');
  childWindow.document.write('Beispiel: <i>foto0001.jpg; Bildtext</i>.<br>');
  childWindow.document.write('-----------------------------------------------------------<br>');
  childWindow.document.write('Ein Mausklick in ein Bild erzeugt<br>');
  childWindow.document.write('verschiedene Bildübergänge.<br>');
  childWindow.document.write('Ein Mausklick in die Anzeige des<br>');
  childWindow.document.write('Bildnamens schaltet den Hintergrund-<br>');
  childWindow.document.write('sound <b>{backsound.mp3}</b> ein/aus.<br>');
  childWindow.document.write('</body></html>');
}

function changeScreen(scr) {
// ändert die Bildgröße für Normalscreen und Fullscreen
   if (!scr) {
      var h = parseInt(80 * screen.height / 100);
      var data = document.getElementById("menu").getBoundingClientRect();
      var h1 = data.height + 20;
      document.getElementById("pict").style.height = (h - h1) + "px";
      document.getElementById("pict1").style.height = (h - h1) + "px";
   }
   if (scr) {
      var h = parseInt(95 * screen.height / 100);
      var data = document.getElementById("menu").getBoundingClientRect();
      var h1 = data.height + 20;
      document.getElementById("pict").style.height = (h - h1) + "px";
      document.getElementById("pict1").style.height = (h - h1) + "px";
   }
}

document.addEventListener("DOMContentLoaded", function() {
// Anfangsinformation
   if (screen.width < 800) { smallScr = true; }
   else { smallScr = false; }
   if (smallScr) {
      document.getElementById("menu").style.margin = "0px";
      // alert(' Bild auf Minimum verkleinern \n und dann weiter . . . ');
   }
```

```
    document.getElementById("input").addEventListener("change", fileOutput, false);
    bild = document.getElementById("pict");
    backstop();
    var h = parseInt(80 * screen.height / 100);
    var data = document.getElementById("menu").getBoundingClientRect();
    var h1 = data.height + 20;
    document.getElementById("pict").style.height = (h - h1) + "px";
    document.getElementById("pict1").style.height = (h - h1) + "px";
    if (smallScr) { setPosition("pict",0,h1); }
    setPosition("pict1",getPosition("pict").x,getPosition("pict").y);
    document.getElementById("fname").value = "";
    if (smallScr) { setPosition("info",20,h1); }
    if (!smallScr) { setPosition("info",getPosition("menu").x,getPosition("pict").y); }
    alert(iText);
});

function fileOutput(event) {
// Dateinamen erfassen
  var dateien = event.target.files;
  datNames = '';
  for (var i = 0, datei; datei = dateien[i]; i++) {
        datNames = datNames + datei.name + '\n';  // ev. auch datei.size
  }
  var erg = confirm('Dateinamen in <archiv.txt> download ?');
  if (erg) { saveText(); }
}

function saveText() {
// Dateinamen speichern
  fname = 'archiv.txt';
  var textFile = datNames;
  var textBlob = new Blob([textFile], {type:'text/plain'});
  var link = document.createElement('a');
  link.href = window.URL.createObjectURL(textBlob);
  // fname = prompt('Dateiname mit Extension',fname);
  link.download = fname;
  document.body.appendChild(link);
  link.click();
  document.body.removeChild(link);
}

function createList(list,max) {
// trennt die Bildnformationen (blist) von den Bildnamen (flist)
    for (i = 0; i < max; i++) {
      s = list[i];
      pos = s.indexOf(';');
       if (pos == -1) {
         flist[i] = s.trim();
         blist[i] = '?';
      }
      else {
         flist[i] = s.substr(0,pos).trim();
         blist[i] = s.substr(pos+1).trim();
      }
//    alert(flist[i] + ' / ' + blist[i]);
  }
}

function loadFile() {
// Textdatei "archiv.txt" laden
   document.getElementById("pict").style.visibility = "visible";
   document.getElementById("pict1").style.visibility = "visible";
   bild = document.getElementById("pict");
   document.getElementById("satz").style.visibility = "hidden";
   document.getElementById("input").style.visibility = "hidden";
   var input1 = document.getElementById("input1").files[0];
   textType = /text.*/;
   if (!input1.type.match(textType)) { alert('Keine Textdatei !'); return; }
   var reader1 = new FileReader();
   reader1.readAsText(input1,"ISO-8851-1");
   reader1.onload = function(event) {
      var text = event.target.result;
      var archiv = input1.name;
      if (archiv != 'archiv.txt') { alert('Keine <archiv.txt>-Datei !'); return; }
      var txt = text.replace(/\n/gi,',');
      var len = txt.length;
      var stxt = txt.substring(0,len-1);
      alist = stxt.split(',');
      max = alist.length;
      blist = stxt.split(',');
      flist = stxt.split(',');
```

```
      createList(alist,max);
      start = true;
      num = -1;
      GotoNext()
  }
}

function selectFile() {
// Bilddateien auswählen
  document.getElementById("pict").style.visibility = "visible";
  document.getElementById("pict1").style.visibility = "visible";
  document.getElementById("satz1").style.visibility = "hidden";
  document.getElementById("input1").style.visibility = "hidden";
  var fileInput = document.getElementById("input");
  flist = fileInput.files;
  max = flist.length;
  num = -1;
  GotoNext();
}

function readFile(data) {
// Bilddateien einlesen
  even = !even;
  bild = document.getElementById("pict");
  var reader = new FileReader();
  reader.readAsArrayBuffer(data);
  reader.onload = function() {
    let buffer = reader.result;
    let blob = new Blob([new Uint8Array(buffer)]);
    let url = window.URL.createObjectURL(blob);
    bild.src = url;
    if (even)  { pictA.src = url;  }
    if (!even) { pictB.src = url;  }
  }
}

function getPosition(element) {
// Position eines Elements ermitteln
  var el = document.getElementById(element);
  var xPos = 0;
  var yPos = 0;
  while (el) {
    if ((el.tagName == "body") || (el.tagName == "BODY")) {
      var xScroll = el.scrollLeft || document.documentElement.scrollLeft;
      var yScroll = el.scrollTop || document.documentElement.scrollTop;
      xPos += (el.offsetLeft - xScroll + el.clientLeft);
      yPos += (el.offsetTop  - yScroll + el.clientTop);
    }
    else {
      xPos += (el.offsetLeft - el.scrollLeft + el.clientLeft);
      yPos += (el.offsetTop  - el.scrollTop + el.clientTop);
    }
    el = el.offsetParent;
  }
  var position = new Object();
  position.x = xPos;
  position.y = yPos;
  return position;
}

function setPosition(element,x,y) {
// Position eines Elements setzen
   var el = document.getElementById(element);
   el.style.position = "absolute";
   el.style.left = x + "px";
   el.style.top =  y + "px";
}

function centerHor(elem) {
// Ein Element horizontal zentrieren
   el = document.getElementById(elem);
   w = el.offsetWidth;
   h = el.offsetHeight;
   el.style.left = window.innerWidth / 2 - w / 2 + "px";
   //  el.style.top = window.innerHeight / 2 - h /2 + "px";
}

function Opacity(element, percent) {
// Transparenz eines Elements setzen
   var el = document.getElementById(element);
   el.style.opacity = percent / 100;
}
```

```
function Fade(element, start, end, msecs) {
// Fading durchfuehren
   var el = document.getElementById(element);
   var INTVAL = 10;
   el.FADE_Start = start;
   el.FADE_Level = start;
   el.FADE_End  = end;
   var stepval = Math.abs(start - end) / (msecs / INTVAL);
   el.FADE_Step = end > start ? stepval : -stepval;
   el.FADE_Fun  = setInterval(runFade, INTVAL);
   function runFade() {
      el.FADE_Level += el.FADE_Step;
      if (el.FADE_Level >= Math.max(el.FADE_Start, el.FADE_End) ||
          el.FADE_Level <= Math.min(el.FADE_Start, el.FADE_End)) {
        el.FADE_Level = el.FADE_End;
        clearInterval(el.FADE_Fun);
      }
      Opacity(element, el.FADE_Level);
   }
}

function FadeOut(element, msecs) {
// Element ausblenden
   Fade(element, 100, 0, msecs);
}

function FadeIn(element, msecs) {
// Element einblenden
   Fade(element, 0, 100, msecs);
}

function GetFade() {
// Blendenzeit der Fadingblende eingeben
  fadeRun = !fadeRun;
  if (!fadeRun) {
     alert('Fading-Blende aus !');
     document.pict1.style.visibility='hidden';
     fadeRun = false;
     return;
  }
  if (fadeRun) {
    text = 'Blendenzeit zwischen 100 und 5000 MSec \n (kleiner als die Darbietungszeit) \n'
    zahl = prompt(text,fadeTime);
    if (isNaN(zahl)) { zahl = fadeTime0; }
    if ((zahl < 100) || (zahl > 5000)) { zahl = fadeTime0; }
    fadeTime = zahl;
    document.getElementById("pict1").style.visibility='visible';
    document.getElementById("pict").style.visibility='visible';
    document.getElementById("pict1").style.opacity = 0;
    document.getElementById("pict").style.opacity = 1;
  }
}

function Animation() {
// Bildübergang auswählen
  if (!okay) { return; }
  var s = '0 = KEINE Blende bei Bildwechsel\n' +
          '1 = Dimming, 2 = Zooming\n' +
          '3 = Flying left, 4 = right\n' +
          '5 = Moving left, 6 = up\n' +
          '7 = 3D-Flipping, 8 = zufällig\n' +
          '9 = FADING-Blende (ein/aus)\n\n';
  var zahl = prompt(s,animat);
  if (!isNaN(zahl)) {
    if ((zahl < 0) || (zahl > 9)) { zahl = 0; }
  }
  else { zahl = 0; }
  if ( zahl == 9 ) { animat=0; GetFade(); return;}
  animat = zahl;
}

function doAnimate(x) {
// Verschiedene Bildübergänge realisieren
   if (x == 0) { return; }
   if (x == 8) { var n = Math.floor(blenden * Math.random()) + 1; }
   if ((x >= 1) && (x <= 7)) { var n = x; }
   if (n == 1) {  // Dimming
     bild.animate([
       { opacity: '0'},
       { opacity: '1'}
     ],{ duration: antime}
     );
   }
```

```
    if (n == 2) {  // Zooming
      bild.animate([
        { transform: 'scale(0.25)'},
        { transform: 'scale(1.50)'},
        { transform: 'scale(1.0)'}
      ],{ duration: 1.5*antime}
      );
    }
    if (n == 3) {  // Flying from left
      bild.animate([
        { transform: 'translateX(-250px)'  + 'rotate(45deg)'},
        { transform: 'scale(0.5)'},
        { transform: 'scale(1.0)'}
      ],{ duration: antime}
      );
    }
    if (n == 4) {  // Flying from right
      bild.animate([
        { transform: 'translateX(250px)'  + 'rotate(45deg)'},
        { transform: 'scale(0.5)'},
        { transform: 'scale(1.0)'}
      ],{ duration: antime}
      );
    }
    if (n == 5) {  // Moving from left
      bild.animate([
        { opacity: '0'},
        { marginLeft: '-100%'},
        { opacity: '0'},
        { marginLeft: '0%'},
        { opacity: '1'}
      ],{ duration: antime}
      );
    }
    if (n == 6) {  // Moving from down
      bild.animate([
        { opacity: '0'},
        { marginTop: '100%'},
        { opacity: '0'},
        { marginTop: '0%'},
        { opacity: '1'}
      ],{ duration: antime}
      );
    }
    if (n == 7) {  // 3D-Flipping
      bild.animate([
        { origin: '50% 50%'},
        { transform: 'perspective(600px)' + 'rotateY(180deg)'},
      ],{ duration: antime}
      );
    }
}

function GotoImage()  {
// Sprung zu einem Bild
  if (!okay) { return; }
  if (ShowRun) { return; }
  var zahl = prompt('Nummer der Bildseite von 0 bis '+(max-1),num);;
  if (!isNaN(zahl)) {
    num = zahl - 1;
    if (num < 0) { num = -1; }
    if (num > (max-1)) { num = max-1; }
  }
  else { num = -1; }
  num = 1 * num;
  GotoNext();
}

function GotoNext() {
// nachfolgendes Bild
  num = num + 1;
  if (num > max-1) { num = 0; }
  even = !even;
  file = flist[num];
  if (start) {
     document.getElementById("fname").value = num + '/' + max + ' (' + file + ')';
     pict.src = file;
     bild.src = file;
     doAnimate(animat);
  }
```

```
    if (!start) {
       document.getElementById("fname").value = num + '/' + max + ' (' + flist[num].name + ')';
       readFile(file);
       pict.src = file;
       bild.src = file;
       doAnimate(animat);
    }
    if (!fadeRun) { document.getElementById("pict1").style.visibility = "hidden"; }
    if (fadeRun && (num > 0)) {
       if (start) {
          document.getElementById("pict1").style.top = document.getElementById("pict").style.top;
          document.getElementById("pict1").style.visibility = "visible";
          document.getElementById("pict1").style.opacity = 1;
          document.getElementById("pict").style.opacity = 0;
          pictA.src = flist[num-1];
          document.getElementById("pict1").src = pictA.src;
          if (!smallScr) { centerHor("pict1"); }
          pictB.src = flist[num];
          document.getElementById("pict").src = pictB.src;
          if (!smallScr) { centerHor("pict"); }
          FadeOut('pict1',fadeTime);
          FadeIn('pict',fadeTime);
       }
       if (!start) {
          document.getElementById("pict1").style.top = document.getElementById("pict").style.top;
          document.getElementById("pict1").style.visibility = "visible";
          document.getElementById("pict1").style.opacity = 1;
          document.getElementById("pict1").src = pictA.src;
          if (!smallScr) { centerHor("pict1"); }
          FadeOut("pict1",fadeTime);
          document.getElementById("pict").style.opacity = 0;
          document.getElementById("pict").src = pictB.src;
          if (!smallScr) { centerHor("pict"); }
          FadeIn("pict",fadeTime);
       }
    }
    if (start) {
       if (!info) { document.getElementById("info").style.visibility = "hidden"; }
       if (info) {
          s = blist[num];
          if (s == '?') { document.getElementById("info").style.visibility = "hidden"; }
          else {
            document.getElementById("info").innerHTML = blist[num];
            if (!smallScr) { centerHor("info"); }
            document.getElementById("info").style.zIndex = "10";
            document.getElementById("info").style.visibility = "visible";
          }
       }
    }
    okay = true;
    if (ShowRun) { active = window.setTimeout("GotoNext()",zeit);  }
}

function GotoLast() {
// vorangehendes Bild
    if (!okay) { return; }
    if (ShowRun ) { return; }
    num = num - 1;
    even = !even;
    if (num < 0) { num = max-1; }
    file = flist[num];
    if (start) {
       document.getElementById("fname").value = num + '/' + max + ' (' + file + ')';
       pict.src = file;
       bild.src = file;
       doAnimate(animat);
    }
    if (!start) {
       document.getElementById("fname").value = num + '/' + max + ' (' + flist[num].name + ')';
       readFile(file);
       pict.src = file;
       bild.src = file;
       doAnimate(animat);
    }
    if (!fadeRun) { document.getElementById("pict1").style.visibility = "hidden"; }
    if (fadeRun) {
       if (start) {
          document.getElementById("pict1").style.top = document.getElementById("pict").style.top;
          document.getElementById("pict1").style.visibility = "visible";
          document.getElementById("pict1").style.opacity = 1;
          document.getElementById("pict").style.opacity = 0;
          pictA.src = flist[num+1];
```

```
        document.getElementById("pict1").src = pictA.src;
        if (!smallScr) { centerHor("pict1"); }
        FadeOut('pict1',fadeTime);
        pictB.src = flist[num];
        document.getElementById("pict").src = pictB.src;
        if (!smallScr) { centerHor("pict"); }
        FadeIn('pict',fadeTime);
    }
     if (!start) {
        document.getElementById("pict1").style.top = document.getElementById("pict").style.top;
        document.getElementById("pict1").style.visibility = "visible";
        document.getElementById("pict1").style.opacity = 1;
        document.getElementById("pict1").src = pictB.src;
        if (!smallScr) { centerHor("pict1"); }
        FadeOut("pict1",fadeTime);
        document.getElementById("pict").style.opacity = 0;
        document.getElementById("pict").src = pictA.src;
        if (!smallScr) { centerHor("pict"); }
        FadeIn("pict",fadeTime);
     }
  }
  if (start) {
     if (!info) { document.getElementById("info").style.visibility = "hidden"; }
     if (info) {
        s = blist[num];
        if (s == '?') { document.getElementById("info").style.visibility = "hidden"; }
        else {
          document.getElementById("info").innerHTML = blist[num];
          if (!smallScr) { centerHor("info"); }
          document.getElementById("info").style.zIndex = "10";
          document.getElementById("info").style.visibility = "visible";
        }
     }
  }
}

function PrintImage() {
// Bild drucken
  if (!okay) { return; }
  print();
}

function ShowStart() {
// Automatische Bildershow starten
  if (!okay) { return; }
  if (ShowRun) { return; }
  ShowRun = true;
  var zahl = prompt('Darbietungszeit zwischen 1000 und 9000 MSec',zeit);
  if (!isNaN(zahl)) {
    if ((zahl < 1000) || (zahl > 9000)) { zahl = 5000; }
  }
  else { zahl = 5000; }
  zeit = zahl;
  active = window.setTimeout("GotoNext()",zeit);
 }

function ShowStop() {
// Automatische Bildershow beenden
  if (!okay) { return; }
  ShowRun = false;
  window.clearTimeOut(active);
}

backstop = function() {
// Hintergrund-Sound "backsound.mp3" ausschalten
  document.getElementById('bplay').pause();
}

backplay = function() {
// Hintergrund-Sound "backsound.mp3" einschalten
  document.getElementById('bplay').play();
}

function soundOnOff() {
// Hintergrund-Sound "backsound.mp3" ein/aus
   snd = document.getElementById('bplay');
   sound = !sound;
   if (!sound) { backstop(); }
   if (sound) { backplay(); }
}
```

```
   function enterFullscreen(element) {
   // Fullscreen einstellen
     if (!okay) { return; }
     if (smallScr) { return; }
     if(element.requestFullscreen) { element.requestFullscreen(); }
     else if(element.mozRequestFullScreen) { element.mozRequestFullScreen(); }
     else if(element.msRequestFullscreen) { element.msRequestFullscreen(); }
     else if(element.webkitRequestFullscreen) { element.webkitRequestFullscreen(); }
     fullScr = true;
     changeScreen(fullScr);
   }

   function quitFullscreen() {
   // Fullscreen verlassen
     if (!okay) { return; }
     if (smallScr) { return; }
     if(document.exitFullscreen) { document.exitFullscreen(); }
     else if(document.mozCancelFullScreen) { document.mozCancelFullScreen();  }
     else if(element.msCancelFullscreen) { element.msCancelFullscreen(); }
     else if(document.webkitExitFullscreen) { document.webkitExitFullscreen(); }
     fullScr = false;
     changeScreen(fullScr);
   }

   function infoText() {
   // optionale Bildinformation anzeigen (ein/aus)
     if (!start) { return; }
     info = !info;
     if (info) { alert('Bildtext-Anzeige eingeschaltet !'); }
     if (!info) { alert('Bildtext-Anzeige ausgeschaltet !'); }
   }
</script>
</head>

<body>
<div id = "menu">
<input type = "button" class = "btn" value = "Next" onclick = "GotoNext()">
<input type = "button" class = "btn" value = "Back" onclick = "GotoLast()">
<input type = "button" class = "btn" value = "Goto" onclick = "GotoImage()">
<input type = "button" class = "btn" value = "Start" onclick = "ShowStart()">
<input type = "button" class = "btn" value = "Stop" onclick = "ShowStop()">
<input type = "button" class = "btn" value = " ? " onclick = "showHelp()">
<br>
<input type = "button" class = "btn" value = " + " onclick =
                                                  "enterFullscreen(document.documentElement)">
<input type = "button" class = "btn" value = " - " onclick = "quitFullscreen()">
<input type = "text" id="fname" value="" size="33" onmousedown="soundOnOff()" readonly>
<input type = "button" class = "btn" value = " i " onclick = "infoText()">
<br>
<span id="satz">
Ordner <input type="file" multiple id="input" accept=".jpg,.png" onchange = "selectFile()">
</span><br>
<span id="satz1">
Archiv <input type="file" id="input1" accept=".txt" onchange = "loadFile()">
</span><br>
</div>
<div id = "info">
</div
<br><br>
<img src="" id="pict" style="visibility:hidden" onmousedown="Animation()">
<img src="" id="pict1" style="visibility:hidden" onmousedown="Animation()">
<audio id="bplay" name="bplay" autoplay loop><source src="backsound.mp3" type="audio/mp3"></audio>
</body>
</html>
```

Erklärungen

(1) Im ersten HTML-Tag **<input type =“file“ id=“input“ . . .>** ermöglicht JavaScript einen lesenden Zugriff auf Dateien und auf einschlägige Datei-Informationen wie Name, Größe und Typ. Das Attribut „**multiple**“ erlaubt die Auswahl und Eingabe von mehreren Werten. Mit dem Attribut „**files**“ wird das Array „**flist**“ erstellt. Das erfolgt in der Funktion „**selectFile()**“, welche als „**onchange**“-Eventhandler dem HTML-Tag angehängt ist. In einem Fenster können ein Ordner des Computers und dort die Dateien ausgewählt werden. Deren Namen sind als nummerierte Items im „**flist**“-Array gespeichert. Die Anzahl der Dateien liefert die Methode „**flist.length**“.

Als Zusatz kann im Eventhandler die Routine „**fileOutput()**“ aufgerufen werden, die alle Dateinamen in eine Textdatei „**archiv.txt**“ in den Download-Ordner abspeichert. Diese Archivdatei kann dann in den aktuellen Bildordner kopiert und dort weiter verarbeitet werden.

(2) Im zweiten HTML-Tag **<input type =“file“ id=“input1“ . . .>** ermöglicht JavaScript einen lesenden Zugriff auf eine Textdatei „**archiv.txt**“ im aktuellen Ordner. Diese Datei enthält in ihren Textzeilen die Namen der Bilddateien. Optional kann daneben - durch einen Strichpunkt (;) getrennt - noch eine kurze Bildinformation stehen, welche im entsprechenden Eventhandler „**loadFile()**“ durch die Funktion „**createList(list,max)**“ von der Namensliste getrennt wird. Das liefert dann das Namens-Array „**flist**“ und das Array „**blist**“ der Bildinformationen.

Im ersten Fall (1) bei einer Dateiauswahl aus einem beliebigen Ordner wird die Steuervariable „**start**“ auf false gesetzt. Im zweiten Fall (2) beim Einlesen einer Archivdatei „**archiv.txt**“ aus dem aktuellen Ordner wird die Steuervariable „**start**“ auf true gesetzt.

(3a) In der Funktion „**GotoNext()**“ wird beim ersten Fall (1) die wichtige Routine „**readFile()**“ mit dem Parameter „**file**“ als eine Datei aus dem „**flist**“-Array aufgerufen. In dieser Routine wird eine neue Instanz des so genannten „**FileReade**r“-Objektes erzeugt. Für die mit „**file**“ bezeichnete Datei ist ein Array-Buffer reserviert, der beim „**onload**“-Event in einen „**Blob**“ gespeichert wird, der die einzelnen Bytes der Datei enthält. Zu dem Blob wird dann ein entsprechender Adresszeiger (**URL**) erzeugt. Zuletzt wird einem **Image**-Objekt diese Daten-URL als Quelle (**source**) zugewiesen. Dadurch wird die Bilddatei als Image am Bildschirm dargestellt.

(3b) In der Funktion „**GotoNext()**“ wird beim zweiten Fall (2) die mit „**file**“ bezeichnete Datei direkt aus dem „**flist**“-Array entnommen. Zusätzlich kann die optionale Bildinformation aus dem Array „**blist**“ mit dem Wechselschalter **[i]** am Bildschirm angezeigt oder nicht angezeigt werden.

(4) Mit einem Mausklick in das aktuelle Bild werden mithilfe der Funktion „**Animation()**“ verschiedene Bildübergänge (**x**) ausgewählt, die dann in der Funktion „**doAnimate(x)**“ realisiert sind. Aktuell stehen sieben unterschiedliche Bildübergänge (x = 1, 2, … 7) zur Auswahl. Bei x = 0 erfolgt keine Animation, bei x = 8 hingegen wird der Übergang zufällig bestimmt.

Zusätzlich und unabhängig von diesen Bildübergängen kann mit x = 9 abwechselnd eine Fading-Blende ein- und ausgeschaltet werden.

(5) Mit einem Mausklick in die Anzeige des Bildnamens kann eine Hintergrundmusik „**backsound.mp3**“ ein- und ausgeschaltet werden, wenn sie im aktuellen Ordner existiert.

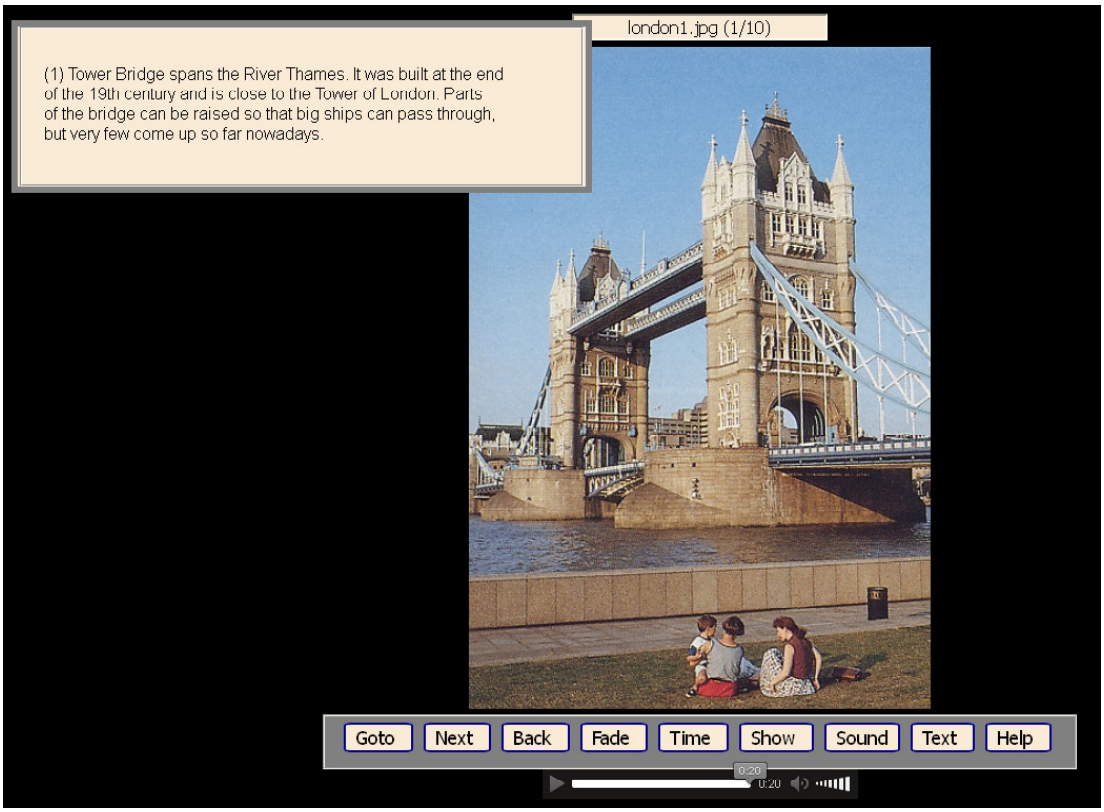
(6) Mit den Schaltern [+] und [-] kann die Bilddarstellung zwischen Fullscreen und Normalscreen gewechselt werden („***enterFullscreen(document.documentElement)***“ und „***quitFullscreen()***“).

(7) Mit dem Wechselschalter [?] wird ein Hilfstext ein- und ausgeblendet.

(8) Die restlichen Funktionen des Programms **GotoLast()**, **GotoImage()**, **ShowStart(), ShowStop(), . . .** sind selbsterklärend.

(7.3.1) Eine Multimedia-Show

Mit dem Programm „**paushow.html**“ kann eine Multimediashow mit Bildern, Texten, Sounds und Videos erzeugt werden. Die Grafik zeigt ein Bild aus einer Show über London. Die verschiedenen Menüfunktionen der Menüleiste können über die Schalter oder über die Tastatur ausgeführt werden.



Menüfunktionen mit Schaltern und Tastaturzeichen:

<GOTO>,<g> Bild anspringen
<NEXT>,<n> Bild vorwärts
<BACK>,<b> Bild rückwärts
<FADE>,<f> Blenden ein/aus
<TIME>,<d> Bild-Darbietungszeit
<SHOW>,<s> Slideshow e/a
<SOUND>,<o> falls vorhanden e/a
<TEXT>,<t> falls vorhanden e/a
(Text mit Maus verschiebbar)
<HELP>,<h> Hilfe e/a

<MOUSE-Left> Slideshow e/a
<MOUSE-Right>,<m> Menübar e/a
<+>,<->,<x> Zoom (in, out, off)
<0>,<1>,<2> Helligkeit ändern
<c> FullScreen e/a
<Esc> FullScreen aus

Der folgende Text zeigt den gesamten Inhalt der externen JavaScript-Datei „**bilder.js**“ einer individuellen Multimediashow über London. Das ist eine reine Textdatei, welche alle wichtigen Steuervariablen für das Programm „**paushow.html**“ enthält. Sie kann mit einem Texteditor erstellt werden, wodurch das Programm individuell konfiguriert wird und für beliebige Bildershows verwendet werden kann.

Inhalt der JavaScript-Datei „**bilder.js**“:

```
// 12 externe Parameter für "paushow.html"
bildtitel     = "London 1966";
bildname    = "london";
bildtyp       = ".jpg"
bildanzahl  = 16;
bildnullen   = false;
bildsound   = true;
soundtyp    = ".mp3";
bildvideo = true;
videotyp = ".mp4";
videonamen = "london13.mp4, london15.mp4";
bildtext =  true;
textinhalt =
"(1) <b>Tower Bridge</b> spans the River Thames. It was built at the end#"+
"of the 19th century and is close to the Tower of London. Parts#"+
"of the bridge can be raised so that big ships can pass through,#"+
"but very few come up so far nowadays.# §"+
"(2) The <b>Tower of London</b> was built as a Norman fortress in the#"+
"11th century. It was also the residence of British kings, a prison#"+
"in which famous people were killed , and a place where arms#"+
"were kept. Today it is a museum housing the Crown Jewels.# §"+
"(3) The beautiful building of the <b>Houses of Parliament</b> is close to#"+
"the River Thames. It was opened in 1852 after the old palace had#"+
"been almost totally destroyed by a fire. There are two Houses,#"+
"the House of Commons and the House of Lords. Laws are made#"+
"in the House of Commons. The famous clocktower of the Houses#"+
"of Parliament is often called «Big Ben». Big Ben, however, is the#"+
"name of the bell in the clocktower.# §"+
"(4) The Queen and her family live in <b>Buckingham Palace</b>. There#"+
"are more than six hundred rooms in this famous palace. When#"+
"the Queen is at home, you can see a flag on the roof of the palace.# §"+
"(5) <b>Downing Street 10</b> is the address of the office of the British#"+
"Prime Minister. Famous politicians meet there and negotiate#"+
"important issues and plan the future of the country. The police-#"+
"man at the entrance door is called «Bobby».# §"+
"(6) <b>Westmister Abbey</b> is London's traditional church for coronations#"+
"and royal weddings. The first king to be crowned at this place in#"+
"1066 was William the Conqueror. In the 13th century the church#"+
"was rebuilt in Gothic style and resembles French cathedrals.# §"+
"(7) <b>St. Paul's Cathedral</b> was built by Sir Christopher Wren. With#"+
"its 111-metre dome the church resembles St. Peter's in Rome.#"+
"The cathedral is the biggest church of the British metropolis.#"+
"The Whispering Gallery alongside the gigantic dome shows the#"+
"church's perfect architecture. A whispered word can be clearly#"+
"understood at the opposite side.# §"+
"(8) <b>Trafalgar Square </b> is a huge square with two beautiful fountains in#"+
"the centre of London. In the middle of the square is Nelson's Column#"+
"Lord Nelson was the famous British admiral who defeated Napoleon#"+
"at the Battle of Trafalgar in Spain in 1805.# §"+
"(9) Londoners often call <b>Piccadilly Circus</b> «Hub of Empire», which#"+
"means the centre of the British Empire. Piccadilly Circus is a#"+
"busy, totally crowded square where five important streets meet.#"+
"In the middle of the place there is a column with the statue of#"+
"Eros on it. In the neighbourhood of Piccadilly Circus there are#"+
"lots of theatres, cinemas, restaurants and nightclubs.# §"+
"(10)<b>Map of LONDON</b># §" +
"(11)<b>Railway to ASCOT</b># §" +
"(12)<b>Visitors' area in ASCOT</b># §" +
"(13)<b>Horse race in ASCOT</b># §" +
"(14)<b>Young beauties in ASCOT</b># §" +
"(15)<b>Cheerful people in ASCOT</b># §" +
"(16)<b>Railway back to LONDON<b># §";
```

Der Name der Bilddateien muss folgenden Aufbau haben: ***bildname****Nummer*.***bildtyp*** (z.B. *foto012.jpg*). Dabei können die Nummern führende Nullen enthalten (***bildnullen = true***) oder nicht (***bildnullen = false***). Im zweiten Falle muss eine Bilddatei beispielsweise statt *foto012,jpg* nur *foto12.jpg* heißen. Die Anzahl der Bilder (***bildanzahl***) kann maximal 999 sein. Jede Show kann auch mit einer Hintergrundmusik ausgestattet werden. Diese muss immer „***backsound.mp3***“ heißen.

Mit dem Schalter [Show] wird eine automatisch ablaufende Bildershow gestartet. Dabei kann vor dem Start die Darbietungszeit der einzelnen Bilder angegeben werden (zeitgesteuerte Show). Gestartet und beendet wird eine Show mit einem linken Mausklick in das angezeigte Bild oder mit der Taste „**s**“.

Grundsätzlich kann zu jeder Bilddatei eine synchrone mp3-Sounddatei abgespielt werden. Die beiden synchronisierten Dateien müssen den gleichen Namen haben, beispielsweise *foto012.jpg* und *foto012.mp3*. Dann muss der Parameter ***bildsound*** auf ***true*** gesetzt werden. In diesem Falle wird bei einem Showstart mit [Show] jede Bilddatei solange dargeboten wie die synchrone Sounddatei läuft (soundgesteuerte Show).

Statt Bilder können auch Videos dargestellt werden. Dann werden die folgenden drei Parameter gesetzt: ***bildvideo*** auf ***true***, ***videotyp*** auf ***“ .mp4“***, und ***videonames*** auf ***“ london13.mp4, london15.mp4“***.
Ein Video ersetzt ein Bild, und ***“videoback.jpg“*** ist ein schwarzes Hintergrundbild für ein Video.

Schließlich kann zu jedem Bild auch ein passender Text in einem Textfeld links oben am Bildschirm angezeigt werden. Das wird mit dem Parameter ***bildtext = true*** entschieden. In diesem Falle wird sowohl die Breite als auch die Höhe des Textfeldes automatisch an den jeweiligen Text angepasst.

Der Parameter ***textinhalt*** enthält den angezeigten Text, der so wie im Muster auf der vorigen Seite aufgebaut ist. Jeder zu einem Bild gehörige Textabschnitt besteht aus einem mit (+) verketteten String. Das Steuerzeichen (#) innerhalb eines Strings bewirkt einen Zeilenvorschub. Das Steuerzeichen (§) am Ende eines Strings bewirkt die Zuordnung zu einem Bild. Es separiert die einzelnen Bildtexte in fortlaufende Array-Elemente. Der letzte String muss mit (;) abgeschlossen werden. Die Bildtexte können leer sein, und die Textanzahl kann auch kleiner als die Bildanzahl sein. Im Bildtext selbst können HTML-Codes für den Textstil eingefügt werden (beispielsweise …….).

Mit den Tasten <2>, <1> und <0> kann die Helligkeit der Bilder schrittweise verändert werden.
Mit den Tasten <+>, <-> und <x> ist ein schrittweises Zoomen der Bilder möglich . Mit einem rechten Mausklick in ein Bild oder mit der Taste <m> wird die Menüleiste ein- und ausgeblendet.
Mit [Sound] oder <o> kann zwischen Bildsound und Hintergrundmusik gewechselt werden.
Erwähnenswert ist noch die Taste <c>, die zwischen Normal- und Full-Screen wechselt. (Grundsätzlich sollte immer der Normal-Screen eingestellt sein, weil die verschiedenen Internet-Browser sich in ihrem Full-Screen-Verhalten unterscheiden).

```
<!DOCTYPE html>
<head>
<title>Bildershow</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Multimediashow mit Bild, Ton, Text und Video">
<meta name="author" content="Herbert Paukert">

<style type="text/css">
   body   { background-color:#000000; color:white; overflow-x: hidden;
            font-family:sans-serif; font-size:18px; font-weight:normal;
            text-align: center; }
   .btn   { border: 2px solid darkblue; border-radius: 10%; background-color: antiquewhite;
            color: black; font-size: 18px; }
   #aus   { margin: 6px; font-size: 16px; text-align: center; background-color: antiquewhite; }
   #atext { position: absolute; left:20px; top:20px; background-color: antiquewhite; color: black;
            border: 6px solid gray; z-index: 2; }
   #asatz { font-family: sans-serif; font-size:16px; text-align: left; padding: 10px; }
   #menu  { padding: 3px; background-color: gray; border: 2px solid white; z-index: 2; }
   #splay { margin: 8px; }
</style>

<script src="bilder.js"></script>

<script>
// paushow.html, Version 7.0 (c) Herbert Paukert
// ------------------------------------------------------ 12 externe Variable
   var bildtitel, bildname, bildtyp, bildanzahl, bildnullen, bildsound, soundtyp;
   var bildvideo, videotyp, videonamen, bildtext, textinhalt;
// ------------------------------------------------------- globale Variable
```

```
var scrWidth = screen.width;
var smallWidth = 800;       // Anpassung an Smartphones
var smallScr = false;       // siehe Skript im "body"
if (scrWidth <= smallWidth) { smallScr = true; }

var bmax    = bildanzahl;   // Anzahl der Bilder
var btext   = textinhalt;   // Bildtexte

var bild = new Array(bmax);
if (!bildnullen) {
   for (var i = 1; i <= bmax; i++) { bild[i] = bildname + i + bildtyp; }
}

if (bildnullen) {
   for (var i = 1; i <= bmax; i++) {
     if (i < 10) {s = '000' + i; }
     if (i > 9 && i < 100) {s = '00' + i; }
     if (i > 99 && i < 1000) {s = '0' + i; }
     bild[i] = bildname + s + bildtyp;
   }
}

if (bildtext) { var tex = btext.split("§"); }      // "§" separiert die Bildtexte
var tmax = tex.length;                             // in ein eigenes Array "tex"

var textRun = bildtext;   // Kennvariable für Bildtext
max = bild.length - 1;    // Anzahl der Bilder

var pictA = new Image();  // Erster Bildspeicher (Speicher für Fading)
var pictB = new Image();  // Zweiter Bildspeicher (Standardspeicher)

var splayer = true;       // Soundplayer-Sichtbarkeit
var videoRun = false;     // Erste Kennvariable für Videoplay
var videoDone = false;    // Zweite Kennvariable für Videoplay

var Bright1 = 1.0;        // Helligkeitswert (Standard)
var Bright  = 1.0;        // Helligkeitswert
var sBright = 0.1;        // Helligkeitsdifferenz

var active;               // Hilfsvariable für Shows
var zeit0 = 5000;         // Zeitdauer der Bilder einer Show (Standard)
var zeit = 5000;          // Zeitdauer der Bilder einer Show
var showRun = false;      // Kennvariable für Shows

var fadeTime = 1000;      // Zeitdauer des Fadings
var fadeRun = false;      // Kennvariable für Fading

zoom = 1.0;               // Zoomwert
zoomRun = false;          // Kennvariable für Zooming

var backsnd  = "backsound.mp3";  // Hintergrund-Sound
var soundRun = true;             // Kennvariable für Vordergrund-Sound

var info = '';            // Titeltext der Bilder
var num  = 1;             // aktuelle Bildnummer
var hy0 = 65;             // Bildhöhe bei NormalScreen (%)
var hy1 = 80;             // Bildhöhe bei FullScreen (%)
var hoe;                  // Höhe der Bilder
var fScreen = false;      // Kennvariable für FullScreen
var tMenu = true;         // Sichbarkeit des Menübalkens

var hilfe =
'=== paushow  (c) H.Paukert ===' + '\n' +
' <GOTO>,<g> Bild anspringen      ' + '\n' +
' <NEXT>,<n> Bild vorwärts        ' + '\n' +
' <BACK>,<b> Bild rückwärts       ' + '\n' +
' <TIME>,<d> Bild-Darbietungszeit ' + '\n' +
' <SHOW>,<s> Slideshow ein/aus    ' + '\n' +
' <FADE>,<f> Blenden e/a          ' + '\n' +
' <SOUND>,<o> falls vorhanden e/a ' + '\n' +
' <p> Player-Sichtbarkeit  e/a    ' + '\n' +
' <TEXT>,<t> falls vorhanden e/a  ' + '\n' +
'    (Text mit Maus verschiebbar) ' + '\n' +
' <HELP>,<h> Hilfe e/a            ' + '\n' +
' <MOUSE-Left> Slideshow e/a      ' + '\n' +
' <MOUSE-Right>,<m> Menübar e/a   ' + '\n' +
' <+>,<->,<x> Zoom (in, out, off) ' + '\n' +
' <0>,<1>,<2> Helligkeit ändern   ' + '\n' +
' <z>, Videozeitlupe einstellen   ' + '\n' +
' <c> FullScreen e/a              ' + '\n' +
' <Esc> FullScreen aus            ' + '\n' +
'=== paushow  (c) H.Paukert ===' + '\n ';
```

```
   // Hilfstext anzeigen
   function ShowHelp() { alert(hilfe); }

   // Text trimmen
   function myTrim(x) { return x.replace(/^\s+|\s+$/gm,'') }

// ------------------------------------------------------ Element-Positionierungen

   // Höhen von NormalScreen und FullScreen berechnen
   screenY  = function() {y = Math.round(hy0*screen.height/100); return y; }
   screenYY = function() {y = Math.round(hy1*screen.height/100); return y; }

   function getPosition(element) {
   // Position eines Elements ermitteln
     var el = document.getElementById(element);
     var xPos = 0;
     var yPos = 0;
     while (el) {
       if ((el.tagName == "BODY") || (el.tagName == "body")) {
         var xScroll = el.scrollLeft || document.documentElement.scrollLeft;
         var yScroll = el.scrollTop || document.documentElement.scrollTop;
         xPos += (el.offsetLeft - xScroll + el.clientLeft);
         yPos += (el.offsetTop  - yScroll + el.clientTop);
       }
       else {
         xPos += (el.offsetLeft - el.scrollLeft + el.clientLeft);
         yPos += (el.offsetTop  - el.scrollTop + el.clientTop);
       }
       el = el.offsetParent;
     }
     var position = new Object();
     position.x = xPos;
     position.y = yPos;
     return position;
   }

   function setPosition(element,x,y) {
   // Position eines Elements setzen
      var el = document.getElementById(element);
      el.style.position = "absolute";
      el.style.left = x + "px";
      el.style.top =  y + "px";
   }

// ------------------------------------------------------ Fading-Blende

   function Opacity(element, percent) {
   // Transparenz eines Elements setzen
      var el = document.getElementById(element);
      el.style.opacity = percent / 100;
   }

   function Fade(element, start, end, msecs) {
   // Fading durchführen
      var el = document.getElementById(element);
      var INTVAL = 10;
      el.FADE_Start = start;
      el.FADE_Level = start;
      el.FADE_End  = end;
      var stepval = Math.abs(start - end) / (msecs / INTVAL);
      el.FADE_Step = end > start ? stepval : -stepval;

      el.FADE_Fun  = setInterval(runFade, INTVAL);

      function runFade() {
         el.FADE_Level += el.FADE_Step;
         if (el.FADE_Level >= Math.max(el.FADE_Start, el.FADE_End) ||
             el.FADE_Level <= Math.min(el.FADE_Start, el.FADE_End)) {
           el.FADE_Level = el.FADE_End;
           clearInterval(el.FADE_Fun);
         }
         Opacity(element, el.FADE_Level);
      }
   }

   // Element einblenden
   function FadeIn(element, msecs) { Fade(element, 0, 100, msecs); }

   // Element ausblenden
   function FadeOut(element, msecs) { Fade(element, 100, 0, msecs); }
```

```
   function GetFade() {
   // Blendenzeit der Fadingblende
     if (smallScr) { alert('Fading nicht möglich!'); return; }
     if (fadeRun == true) {
        alert('Fading-Blende aus !');
        document.pic1.style.visibility='hidden';
        fadeRun = false;
        return;
     }
     if (fadeRun == false) {
       text = 'Blendenzeit zwischen 500 und 5000 MSec'
       zahl = prompt(text,fadeTime);
       zahl = myTrim(zahl);
       if (!isNaN(zahl)) {
         if ((zahl < 500) || ( zahl > 5000)) { zahl = 1000; }
       }
       fadeTime = zahl;
       fadeRun = true;
       document.pic1.style.visibility='visible';
       document.pic2.style.visibility='visible';
       document.pic1.style.opacity = 0;
       document.pic2.style.opacity = 1;
     }
   }

// ------------------------------------------------------ Drag & Drop

   function dragElement(elem) {
   // Hauptprogramm mit drei Unterprogrammen
     var pos1 = 0, pos2 = 0, pos3 = 0, pos4 = 0;
     elem.onmousedown = dragMouseDown;

     function dragMouseDown(ev) {
     // Erstes Unterprogramm
       ev.preventDefault();
       pos3 = ev.clientX;
       pos4 = ev.clientY;
       document.onmousemove = elementDrag;
       document.onmouseup = closeDragElement;
     }

     function elementDrag(ev) {
     // Zweites Unterprogramm
       ev.preventDefault();
       pos1 = pos3 - ev.clientX;
       pos2 = pos4 - ev.clientY;
       pos3 = ev.clientX;
       pos4 = ev.clientY;
       elem.style.top = (elem.offsetTop - pos2) + "px";
       elem.style.left = (elem.offsetLeft - pos1) + "px";
     }

     function closeDragElement() {
     // Drittes Unterprogramm
       document.onmousemove = null;
       document.onmouseup = null;
     }
   }

// ------------------------------------------------- Normalscreen und Fullscreen

   function enterFullscreen(element) {
   // Fullscreen einstellen
     if(element.requestFullscreen) { element.requestFullscreen(); }
     else if(element.mozRequestFullScreen) { element.mozRequestFullScreen(); }
     else if(element.msRequestFullscreen) { element.msRequestFullscreen(); }
     else if(element.webkitRequestFullscreen) { element.webkitRequestFullscreen(); }
   }

   function exitFullscreen() {
   // Fullscreen verlassen
     if(document.exitFullscreen) { document.exitFullscreen(); }
     else if(document.mozCancelFullScreen) { document.mozCancelFullScreen();  }
     else if(document.webkitExitFullscreen) { document.webkitExitFullscreen(); }
   }

   function ScreenToggle() {
   // Wechsel zwischen Fullscreen und Normalscreen
     if (videoDone) { return; }
     fScreen = !fScreen;
```

```
     if (fScreen)  {
        enterFullscreen(document.documentElement);
        hoe = screenYY();
        document.pic1.height = hoe;
        document.pic2.height = hoe;
     }
     else {
        exitFullscreen();
        hoe = screenY();
        document.pic1.height = hoe;
        document.pic2.height = hoe;
     }
   }

// ------------------------------------------------------ Sound-Funktionen

   soundplay = function() {
   // Vordergrund-Sound
      if (!bildsound) { return; }
      var el = document.getElementById('splay');
      ein = bild[num].replace(bildtyp,soundtyp);
      el.src = ein;
      el.width="640";
      el.controls="true";
      if (soundRun) { el.play(); }
      else { el.pause(); }
   }

   backplay = function() {
   // Hintergrund-Sound
      var el = document.getElementById('bplay');
      el.src = backsnd;
      if (!soundRun) { el.play(); }
      else { el.pause(); }
   }

   function SoundToggle() {
   // Wechsel zwischen Vordergrund- und Hintergrund-Sound
      if (showRun) { return; }
      var el = document.getElementById('splay');
      soundRun = !soundRun;
      backplay();
      soundplay();
      if (soundRun && bildsound && !smallScr) { el.style.visibility = 'visible'; }
      if (!soundRun) { el.style.visibility = 'hidden'; }
   }

   function PlayerOnOff() {
   // Sichtbarkeit des Soundplayers
      var el = document.getElementById('splay');
      splayer = !splayer;
      if (splayer) { el.style.visibility = 'visible'; }
      if (!splayer) { el.style.visibility = 'hidden'; }
   }

// ------------------------------------------------------ Video-Funktionen

   videostop = function() {
   // Video stop
      el = document.getElementById('vplay');
      el.width="0";
      el.height="0";
      el.src = "";
      el.controls="false";
      document.pic1.height = hoe;
      document.pic2.height = hoe;
      document.getElementById("aus").style.visibility = "visible";
      if (textRun) { document.getElementById("atext").style.visibility = "visible"; }
      if (soundRun && bildsound && !smallScr) { splay.style.visibility = 'visible'; }
      window.scrollTo({top:-20});
   }

   videoplay = function(ein) {
   // Video play
      if (soundRun && bildsound) { splay.style.visibility = 'hidden'; }
      document.getElementById('menu').scrollIntoView();
      splay.pause();
      bplay.pause();
      document.pic1.height = 0;
      document.pic2.height = 0;
      el = document.getElementById('vplay');
      el.src = ein;
```

```
      if (!smallScr && fScreen) {
         el.width  = "900";
         el.height = "675";
      }
      if (!smallScr && !fScreen) {
         el.width  = "720";
         el.height = "540";
      }
      if (smallScr) {
         el.width  = scrWidth+40;
         el.height = Math.round(scrWidth*0.75);
      }
      el.controls="true";
      el.play();
   }

   videospeed = function() {
   // Set Videospeed
      if (!bildvideo) { return; }
      var el = document.getElementById('vplay');
      var s = prompt('VideoSpeed (0.1 - 2.0)','1.0');
      vspeed = 1 * s;
      if ( (vspeed < 0.1) || (vspeed > 2.0) ) { vspeed = 1.0; }
      el.playbackRate = vspeed;
   }

// ------------------------------------------------------ Text-Funktionen

   function ShowText(k) {
   // Bildtext anzeigen
      if (!textRun) { return; }
      var atext = document.getElementById("atext");
      var asatz = document.getElementById("asatz");
      if (k > tmax) { atext.style.visibility = "hidden"; return; }
      t = myTrim(tex[k-1]);
      if (t == "") { atext.style.visibility = "hidden"; return; }
      atext.style.overflow = "auto";
      atext.style.resize = "both";
      atext.style.width = asatz.style.width;
      atext.style.height = asatz.style.height ;
      s = t.replace(/#/gi,'\n');  // "#" bewirkt einen Zeilenvorschub !!!
      atext.style.visibility = "visible";
      asatz.innerHTML = s;
   }

   function TextToggle() {
   // Textanzeige ein und aus
      if (!bildtext) { return; }
      textRun = !textRun;
      if (textRun) {
          atext.style.visibility = "visible";
          ShowText(num);
      }
      else { atext.style.visibility = "hidden"; }
   }

// ------------------------------------------------------ Initialisierungen

   window.onload = function() {
   // Zwei Images übereinanderlegen
     hoe = screenY();
     document.pic1.height = hoe;
     document.pic2.height = hoe;
     setPosition("pic1",getPosition("pic2").x,getPosition("pic2").y);
     pictB.src = bild[num];
     document.pic2.src = pictB.src;
     pictA.src = bild[num];
     document.pic1.src = pictA.src;
     document.pic1.style.visibility='hidden';
     window.scrollTo(0,0);

     // Titelzeile ausgeben
     info = bild[num] + ' (' + num  + '/' + max + ')';
     document.formu.ausgabe.value = info;

     // Menübalken einrichten
     var menu = document.getElementById('menu');
     l = getPosition("but1").x;
     r = getPosition("but9").x;
     w = (r - l) + 120;
```

```
    if (smallScr) { w = scrWidth; }
    menu.style.width = w +"px";
    menu.style.margin = "auto";

    // Text zum Bild ausgeben
    if (!bildtext) { document.getElementById("atext").style.visibility = "hidden"; }
    if (bildtext)  { ShowText(num); }

     // Sound zum Bild ausgeben
    backplay();
    if (soundRun && bildsound) { soundplay(); }
    if (smallScr) { document.getElementById("splay").style.visibility = "hidden"; }
  }

// ------------------------------------------------------ Goto Bild-Seite

  function GotoImage()  {
  // Zu einem Bild springen
    if (showRun ) { return; }
    zahl = prompt('Nummer der Bildseite zwischen 1 und '+ max, num);
    zahl = myTrim(zahl);
    if (!isNaN(zahl)) {
      num = zahl - 1;
      if (num < 1) { num = 0; }
      if (num > max) { num = max-1 ; }
    }
    else { num = 0; }
    GotoNext();
  }

  function GotoNext() {
  // Zum nachfolgenden Bild wechseln
  // oder eine Bildershow ausführen
    num++;
    if (num > max) {
       num = max;
       if ( showRun ) { showRun = false; ShowStop(); }
       return;
    }
    if (smallScr) { document.getElementById("splay").style.visibility = "hidden"; }

    // Video abfragen
    ein = bildname + num + videotyp;
    videoRun = false;
    videoRun = videonamen.includes(ein);
    if (bildvideo && videoRun && !videoDone) {
        // schwarzes Leerbild als Videohintergrund bei "smallScr"
        if (smallScr) {
           pictB.src = 'videoback.jpg';
           document.pic2.src = pictB.src;
        }
        info = ein + ' (' + num  + '/' + max + ')';
        document.formu.ausgabe.value = info;
        if (textRun) { ShowText(num); }
        videoplay(ein);
        if (showRun) { vplay.onended = function() {isVideo();} }
        videoDone = true;
        return;
        // Seiten-Ausstieg bei ausgeführtem Video
    }

    if (bildvideo && videoDone) {
        videostop();
        videoDone = false;
        // Bei vorangegangenem Video wird jetzt der Videoplayer unsichtbar
    }
    info = bild[num] + ' (' + num  + '/' + max + ')';
    document.formu.ausgabe.value = info;
    if (fadeRun) {
       document.pic1.style.opacity = 1;
       document.pic2.style.opacity = 0;
       setPosition("pic1",getPosition("pic2").x,getPosition("pic2").y);
       pictA.src = bild[num-1];
       document.pic1.src = pictA.src;
       FadeOut('pic1',fadeTime);
       pictB.src = bild[num];
       document.pic2.src = pictB.src;
       FadeIn('pic2',fadeTime);
    }
    else {
      pictB.src = bild[num];
      document.pic2.src = pictB.src;
    }
```

```
  if (textRun) { ShowText(num); }
  if (!soundRun) {
  // Hintergrundmusik ist aktiv
     splay.pause();
     bplay.play();
     if (showRun) { active = setTimeout('GotoNext()', zeit0); }
     else { clearTimeout(active); }
  }
  if (soundRun && bildsound) {
  // Hintergrundmusik ist nicht aktiv
     bplay.pause();
     var el = document.getElementById('splay');
     soundplay();
     if (showRun) { el.onended = function() {isSound();} }
  }
}

function isSound() {
// Show mit Soundplaying
   active = setTimeout('GotoNext()', 100);
}

function isVideo() {
// Show mit Videoplaying
   active = setTimeout('GotoNext()', 100);
}

function GotoLast() {
// Zum vorangehenden Bild wechseln
  if (showRun ) { return; }
  num--;
  if (num < 1) { num = 1; return; }
  if (smallScr) { document.getElementById("splay").style.visibility = "hidden"; }

  // Video abfragen
  ein = bildname + num + videotyp;
  videoRun = false;
  videoRun = videonamen.includes(ein);
  if (bildvideo && videoRun && !videoDone) {
      // schwarzes Leerbild als Videohintergrund bei "smallScr"
      if (smallScr) {
         pictB.src = 'videoback.jpg';
         document.pic2.src = pictB.src;
      }
      info = ein + ' (' + num  + '/' + max + ')';
      document.formu.ausgabe.value = info;
      if (textRun) { ShowText(num); }
      videoplay(ein);
      videoDone = true;
      return;
      // Seiten-Ausstieg bei ausgeführtem Video
  }

  if (bildvideo && videoDone) {
      videostop();
      videoDone = false;
      // Bei vorangegangenem Video wird jetzt der Videoplayer unsichtbar
  }
  info = bild[num] + ' (' + num  + '/' + max +')';
  document.formu.ausgabe.value = info;
  if (fadeRun) {
     document.pic1.style.opacity = 1;
     document.pic2.style.opacity = 0;
     setPosition("pic1",getPosition("pic2").x,getPosition("pic2").y);
     pictA.src = bild[num+1];
     document.pic1.src = pictA.src;
     FadeOut('pic1',fadeTime,false);
     pictB.src = bild[num];
     document.pic2.src = pictB.src;
     FadeIn('pic2',fadeTime,false);
  }
  else {
    pictB.src = bild[num];
    document.pic2.src = pictB.src;
  }
  if (textRun) { ShowText(num); }
  if (soundRun && bildsound) { soundplay(); }
}
```

```
// ------------------------------------------------------ SlideShow-Funktionen

   function HideButtons() {
   // Alle Schalter unsichtbar
     document.getElementById("but1").style.visibility = 'hidden';
     document.getElementById("but2").style.visibility = 'hidden';
     document.getElementById("but3").style.visibility = 'hidden';
     document.getElementById("but4").style.visibility = 'hidden';
     document.getElementById("but5").style.visibility = 'hidden';
     document.getElementById("but6").style.visibility = 'hidden';
     document.getElementById("but7").style.visibility = 'hidden';
     document.getElementById("but8").style.visibility = 'hidden';
     document.getElementById("but9").style.visibility = 'hidden';
     document.getElementById("menu").style.visibility = 'hidden';
   }

   function ShowButtons() {
   // Alle Schalter sichtbar
     document.getElementById("but1").style.visibility = 'visible';
     document.getElementById("but2").style.visibility = 'visible';
     document.getElementById("but3").style.visibility = 'visible';
     document.getElementById("but4").style.visibility = 'visible';
     document.getElementById("but5").style.visibility = 'visible';
     document.getElementById("but6").style.visibility = 'visible';
     document.getElementById("but7").style.visibility = 'visible';
     document.getElementById("but8").style.visibility = 'visible';
     document.getElementById("but9").style.visibility = 'visible';
     document.getElementById("menu").style.visibility = 'visible';
   }

   function GetTime() {
   // Darbietungszeit der Bilder in der Show
     if (showRun) { return; }
     zahl = prompt('Bild-Darbietungszeit zwischen 1000 und 9000 MSec \n ',zeit0);
     zahl = myTrim(zahl);
     if (!isNaN(zahl)) {
       if ((zahl < 1000) || (zahl > 9000)) { zahl = 5000; }
     }
     zeit0 = zahl;
   }

   function ShowStart() {
   // automatische Bildershow starten
     alert('Show-START\n' + 'Show-STOP mit Mausklick' + '\n<c> = FullScreen ein/aus');
     if (soundRun && bildsound)  { zeit = 100; }
     if (!soundRun) { zeit = zeit0; }
     HideButtons();
     num = num - 1;
     active = window.setTimeout("GotoNext()",zeit);
   }

   function ShowStop() {
   // automatische Bildershow stoppen
     alert('Show-STOP' + '\n' + '<c> = FullScreen ein/aus');
     ShowButtons();
     textRun = false;
     TextToggle();
     clearTimeOut(active);
   }

   function ShowToggle() {
   // Wechselschalter für Bildershow
     showRun = !showRun;
     if (showRun) { ShowStart(); }
     if (!showRun) { ShowStop(); }
   }

   function MenuToggle() {
   // Wechsel zwischen Menüleiste sichtbar und unsichtbar
      tMenu = !tMenu
      if (tMenu) { ShowButtons(); }
      else { HideButtons(); }
   }

   function mouseEvent(ev) {
   // Eventhandler für Mouse-Button
     var bt = ev.buttons;
     if (bt == 1) { ShowToggle(); }   // linke Maustaste
     if (bt == 2) { MenuToggle(); }   // rechte Maustaste
     if (bt == 4) { return; }         // Maus-Scrolltaste
   }
```

```
// ------------------------------------------------------ Zoom- und Helligkeit-Funktionen

   function ZoomOff() {
   // Bildgröße normalisieren
     zoomRun = false;
     zoom = 1.0;
     document.getElementById("pic1").style.transform = "scale(1.0)";
     document.getElementById("pic2").style.transform = "scale(1.0)";
   }

   function ZoomOut() {
   // Bild verkleinern
     zoomRun = true;
     zoom = zoom - zoom/10;
     document.getElementById("pic1").style.transform = "scale(" + zoom + ")";
     document.getElementById("pic2").style.transform = "scale(" + zoom + ")";
   }

   function ZoomIn() {
   // Bild vergrößern
     zoomRun = true;
     zoom = zoom + zoom/10;
     document.getElementById("pic1").style.transform = "scale(" + zoom + ")";
     document.getElementById("pic2").style.transform = "scale(" + zoom + ")";
   }

   function ChangeBrigthness(x) {
   // Bild normal(0), dünkler(1), heller(2)
      if (x == 0) { Bright = Bright1; }
      if (x == 1) { Bright = 1*Bright - sBright; }
      if (x == 2) { Bright = 1*Bright + sBright; }
      el = document.getElementById("pic2");
      el.style.filter = "brightness(" + 1*Bright + ")";
      el.style.webkitFilter = "brightness(" + 1*Bright + ")";
   }

// ------------------------------------------------------ Tastatur-Abfragen

   function KeyListener(ev) {
   // Eventhandler für Tatatureingaben
     var taste = ev.which || ev.keyCode;
//    alert(taste);
     if (taste == 27)  { if (fScreen) {ScreenToggle();} }  // Esc
     if (taste == 67)  { ScreenToggle(); }          // c
     if (taste == 71)  { GotoImage(); }             // g
     if (taste == 78)  { GotoNext(); }              // n
     if (taste == 66)  { GotoLast(); }              // b
     if (taste == 70)  { fadeRun = !fadeRun; }      // f
     if (taste == 68)  { GetTime(); }               // d
     if (taste == 83)  { ShowToggle(); }            // s
     if (taste == 79)  { SoundToggle(); }           // o
     if (taste == 80)  { PlayerOnOff(); }           // p
     if (taste == 84)  { TextToggle(); }            // t
     if (taste == 72)  { ShowHelp(); }              // h
     if (taste == 171) { ZoomIn(); }                // +
     if (taste == 173) { ZoomOut(); }               // -
     if (taste == 88)  { ZoomOff(); }               // x
     if (taste == 48)  { ChangeBrigthness(0); }     // 0
     if (taste == 49)  { ChangeBrigthness(1); }     // 1
     if (taste == 50)  { ChangeBrigthness(2); }     // 2
     if (taste == 90)  { videospeed(); }            // z
     if (taste == 77)  { MenuToggle(); }            // m
   }

// ------------------------------------------------------ JS-Scriptende
</script>
</head>
```

```
<body id="bd" onkeydown="KeyListener(event)" oncontextmenu = "return false;">
<div id="atext">
   <pre id="asatz"> </pre>
</div>

<form name = "formu">
<audio id="bplay" name="bplay" loop autoplay><source src="" type="audio/mp3"></audio>
<input type="text" id="aus" name="ausgabe" value="" size="30" readonly><br>
<video id="vplay" name="vplay" width="0"><source src=" "></video>
<img src="" id="pic1" name="pic1" height=500 onmousedown="mouseEvent(event)">
<img src="" id="pic2" name="pic2" height=500 onmousedown="mouseEvent(event)">
<div id="menu" name="menu">
<input type = "button" id = "but1"  class = "btn" value = "Goto" onclick = "GotoImage()"> 
<input type = "button" id = "but2"  class = "btn" value = "Next" onclick = "GotoNext()"> 
<input type = "button" id = "but3"  class = "btn" value = "Back" onclick = "GotoLast()"> 
<input type = "button" id = "but4"  class = "btn" value = "Fade" onclick = "GetFade()"> 
<input type = "button" id = "but5"  class = "btn" value = "Time" onclick = "GetTime()"> 
<input type = "button" id = "but6"  class = "btn" value = "Show" onclick = "ShowToggle()"> 
<input type = "button" id = "but7"  class = "btn" value = "Sound" onclick = "SoundToggle()"> 
<input type = "button" id = "but8"  class = "btn" value = "Text" onclick = "TextToggle()"> 
<input type = "button" id = "but9"  class = "btn" value = "Help" onclick = "ShowHelp()"> 
</div>
</form>

<audio id="splay" name="splay" width="0"> <source src="" type="audio/mp3"></audio>
<script>
   // Textfeld mit der Maus verschieben
   if (!smallScr) { dragElement(document.getElementById("atext")); }
   // Einfache Programm-Anpassung an Smartphones
   if (smallScr) {
      document.getElementById("bd").style.textAlign = "left";
      document.getElementById("pic1").style.width = scrWidth + "px";
      document.getElementById("pic1").style.heigth = "auto";
      document.getElementById("pic2").style.width = scrWidth + "px";
      document.getElementById("pic2").style.height = "auto";
      document.getElementById("pic1").style.zIndex = "-1";
      document.getElementById("pic2").style.zIndex = "-1";
      document.getElementById("vplay").style.zIndex = "2";
      document.getElementById("atext").style.zIndex = "3";
      setPosition("aus",15,0);
      setPosition("menu",20,40);
      setPosition("atext",20,120);
      setPosition("pic1",20,120);
      setPosition("pic2",20,120);
      setPosition("vplay",0,120);
   }
</script>
</body>
</html>
```

Zusatzbemerkungen:

Im Programmcode wird auch eine einfache Anpassung an Smartphone-Bildschirme realisiert. Dazu wird am Anfang die aktuelle Bildschirmbreite abgefragt. Wenn sie kleiner oder gleich dem Grenzwert von 800 Pixeln ist, dann wird die Steuervariable "***smallScr***" auf ***true*** gesetzt, sonst auf ***false***.

Wenn "***smallScr***" auf ***true*** gesetzt ist, wird erstens der Programmcode an einigen Stellen modifiziert (z.B. keine Sichtbarkeit des Soundplayers, keine Bild-Fading möglich, usw.), und zweitens wird im "***body***" eine JavaScript-Routine aufgerufen, wo folgende Anpassungen vorgenommen werden:

- Linksbündige Darstellung des gesamten ***body***-Inhalts.
- Die Bildbreiten der beiden Images werden auf die aktuelle Bildschirmbreite gesetzt.
- Die wichtigsten Programmobjekte werden mittels "***setPosition***" neu positioniert.
- Mithilfe von "***zIndex***" werden die Videos VOR den Images dargestellt, und alle Texte VOR den Images und Videos. (Mit dem Wechselschalter [***Text***] können die Texte eingeblendet oder ausgeblendet werden.)

Somit werden mit dem Programm "***paushow.html***" nicht nur auf Desktops, sondern auch auf Smartphones Multimedia-Shows mit Texten, Bildern, Sounds und Videos ausgeführt.

(7.4.1) Grundstufe des Sprachlernens

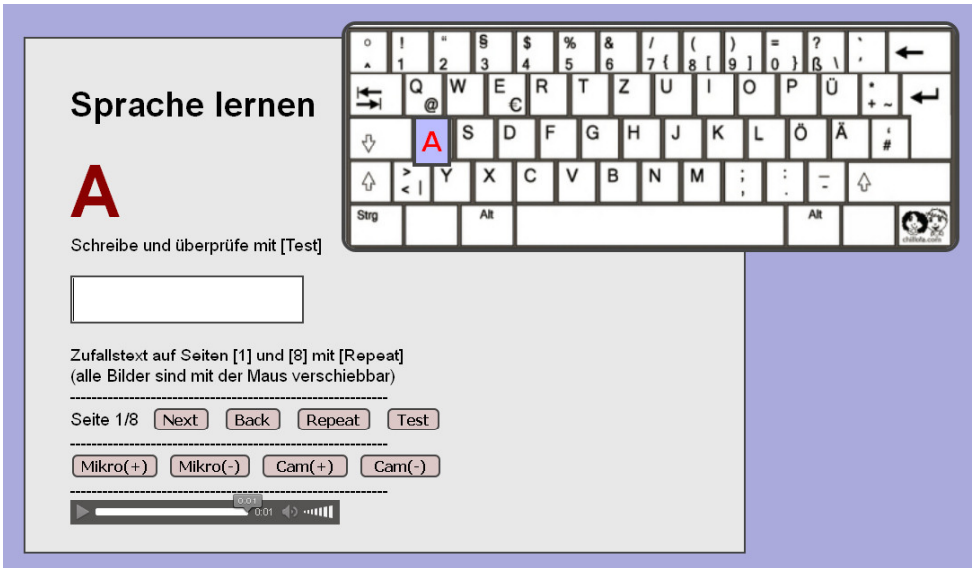
Mit dem Programm "**sprachlern.html**" können Sprachanfänger die ersten Schritte zum Erlernen der deutschen Sprache ausführen. Das Programm ist auch für Vorschulkinder geeignet.

Zu Beginn erscheint ein angeleitetes Lernen durch eine Lehrperson sinnvoll, wobei grundlegende Erklärungen und Hilfen gegeben werden können.

Ziel ist zuerst das Lesen, Nachsprechen und Eintasten der Buchstaben des Alphabets, wobei das Programm automatisch auf Großbuchstaben umstellt. Danach werden einfache Wörter gelesen, nachgesprochen und eingetastet. Zusätzlich kann versucht werden, die Buchstaben und Wörter auf einem Blatt Papier zu schreiben.

Das Programm besteht aus acht Bildseiten, wo Buchstaben oder einfache Wörter am Bildschirm dargestellt werden, begleitet von einer entsprechenden Grafik und einem entsprechenden Sound. Das Nachsprechen erfolgt über ein aufrufbares Mikrofon. Zusätzlich kann mit einer Webkamera aufgenommen werden. Das Schreiben erfolgt mit der Tastatur, wobei das Schreibergebnis immer mit "richtig" oder "falsch" bewertet wird.

Nachfolgend sind die ersten zwei Bildseiten dargestellt.



```
<!DOCTYPE html>
<html>
<head>
<title>spralern.html</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="author" content="Paukert Herbert">
<meta name="description" content="Sprachlernen Grundstufe">
<meta name="keywords" content="Sprachlernen Grundstufe">

<style>
body    {background-color:#AAAADD; color:black; font-family:Arial,sans-serif;
         font-size:19px; font-weight:normal; text-align:left; margin:5%; }
#fs     {width: 750px; background-color: #E8E8E8; padding: 25px 0px 25px 50px;
         border: 2px solid #444444; margin: auto; };
h1      {color: black; }
.cd1    {border: 2px solid #444444; background-color: #FFFFFF; font-size: 40px;}
.cd2    {margin: 2px; border: 2px solid #444444; background-color: #DDC8C8; font-size: 16px;
         font-weight: bold; border-radius: 6px;}
#buch   {color: darkred; font-size: 80px; font-weight:bold; }
#divImg {position: absolute; left:820px; top: 10px; }
#myImg  {width: 680px; border: 4px solid #444444; border-radius: 16px; }
#divZei {width: 40px; height:40px;}
#zei    {width: 40px; border: 4px solid #444444; padding: 6px; color: red; background: #BBBBFF;
         font-family:Arial,sans-serif; font-size: 32px; font-weight: bold;}
</style>

<script>
var max = 8;
var anz = 29;
var letters ="ABCDEFGHIJKLMNOPQRSTUVWXYZÄÖÜ";
var a = "+,MAMA,MAUS,HAUS,TÜR,FENSTER,DACH,*";
var b = "tasten1.jpg,mama.jpg,maus.jpg,haus.jpg,tür.jpg,fenster.jpg,dach.jpg,haus1.jpg";
var c = "+,mama.mp3,maus.mp3,haus.mp3,tür.mp3,fenster.mp3,dach.mp3,*";

anz1 = 6;
var words  = 'GARTEN,WIESE,BAUM,HAUS,TÜR,DACH';
var sounds = 'garten.mp3,wiese.mp3,baum.mp3,haus.mp3,tür.mp3,dach.mp3';

var arr1 = a.split(',');
var arr2 = b.split(',');
var arr3 = c.split(',');

var arr4 = words.split(',');
var arr5 = sounds.split(',');

// A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, Ä, Ö, Ü
var xKoord =
[75,285,187,175,155,214,260,310,388,352,401,446,374,327,435,483,65,203,121,250,342,235,110,143,96,295,
540,493,530];
var yKoord =
[102,150,150,102,52,102,102,102,52,102,102,102,150,150,52,52,52,52,102,52,52,150,52,150,150,52,
105,105,55];

var num = -1;    // Bildseiten-Nummer
var word = '';   // Angezeigte Buchstaben oder Wörter
var bild;        // Bild-Name
var snd;         // Sound-Name
var micro;       // Mikrofon-Variable
var camera;      // Kamera-Variable

// Mikrofon
function NewWindow()  { micro = window.open("paumicro.html","micro","left=810,top=10,
width=500,height=450,toolbar=1,scrollbars=1"); }
function ShutWindow() { micro.close(); }

// Kamera
function NewWindow1()  { camera = window.open("paucamera.html","camera","left=10,top=10;
width=500,height=680,toolbar=1,scrollbars=1"); }
function ShutWindow1() { camera.close(); }

// Soundplayer
soundplay = function(snd) { document.getElementById('splay').src = snd; splay.width="320";
splay.controls="true"; splay.play(); }
soundstop = function(snd) { document.getElementById('splay').src = snd; splay.width="0";
splay.controls="false"; splay.pause(); }

// Stringtrimmer
function myTrim(x)  { return x.replace(/^\s+|\s+$/gm,''); }
```

```
function getPosition(element) {
// Position eines Elements ermitteln
   var el = document.getElementById(element);
   var xPos = 0;
   var yPos = 0;
   while (el) {
     if (el.tagName == "body") {
       var xScroll = el.scrollLeft || document.documentElement.scrollLeft;
       var yScroll = el.scrollTop || document.documentElement.scrollTop;
       xPos += (el.offsetLeft - xScroll + el.clientLeft);
       yPos += (el.offsetTop  - yScroll + el.clientTop);
     }
     else {
       xPos += (el.offsetLeft - el.scrollLeft + el.clientLeft);
       yPos += (el.offsetTop  - el.scrollTop + el.clientTop);
     }
     el = el.offsetParent;
   }
   var position = new Object();
   position.x = xPos;
   position.y = yPos;
   return position;
}

function setPosition(element,x,y) {
// Position eines Elements setzen
   var el = document.getElementById(element);
   el.style.position = "absolute";
   el.style.left = x + "px";
   el.style.top =  y + "px";
}

function dragElement(elem) {
// Verschieben von "div"-Objekten mit der Maus
// Hauptprogramm mit drei Unterprogrammen
   var pos1 = 0, pos2 = 0, pos3 = 0, pos4 = 0;
   elem.onmousedown = dragMouseDown;

   function dragMouseDown(ev) {
   // Erstes Unterprogramm
      document.getElementById('divZei').style.visibility = "hidden";
      ev.preventDefault();
      pos3 = ev.clientX;
      pos4 = ev.clientY;
      document.onmousemove = elementDrag;
      document.onmouseup = closeDragElement;
   }

   function elementDrag(ev) {
   // Zweites Unterprogramm
      ev.preventDefault();
      pos1 = pos3 - ev.clientX;
      pos2 = pos4 - ev.clientY;
      pos3 = ev.clientX;
      pos4 = ev.clientY;
      elem.style.top = (elem.offsetTop - pos2) + "px";
      elem.style.left = (elem.offsetLeft - pos1) + "px";
   }

   function closeDragElement() {
   // Drittes Unterprogramm
      document.onmousemove = null;
      document.onmouseup = null;
   }
}

function pruef(x,y) {
// Prüfroutine der Antwort
  ant0 = '*** FALSCH ***';
  ant1 = '*** RICHTIG ***';
  ant  = '';
  y = document.getElementById('antw').value;
  x = word;
  if (x == y) { ant = ant1; soundplay('erg1.mp3'); }
  else { ant = ant0; soundplay('erg0.mp3'); }
  alert(ant);
  document.getElementById('antw').focus();
//  splay.onended = function() { soundstop(); }
}
```

```
function KeyListener(ev) {
// Tastatur-Event-Handler
  var s = document.getElementById('antw').value;
  document.getElementById('antw').value = s.toUpperCase();
}
function initPage() {
// Startseite mit Initialisierungen
  num = 0;
  s = (num+1) + '/' + (max);
  document.getElementById('num').innerHTML = s;
  if (num == 0) { document.getElementById('myImg').style.width = 680 + 'px'; }
  else { document.getElementById('myImg').style.width = 580 + 'px'; }
  i = Math.floor(anz*Math.random() );
  word = letters.charAt(i);
  bild = arr2[num];
  if (bild == 'no') { bild = 'no.jpg'; }
  snd  = arr3[num];
  if (snd == '+')  { help = word +'.mp3'; snd = help.toLowerCase(); }
  if (snd != 'no') { soundplay(snd); }
  document.getElementById('buch').innerHTML = word;
  document.getElementById('myImg').src = bild;
  // Buchstaben in der Grafik positionieren und anzeigen - Start
     x = getPosition('myImg').x + xKoord[i];
     y = getPosition('myImg').y + yKoord[i] + 8;
     setPosition('divZei',x,y);
     document.getElementById('zei').innerHTML = word;
     document.getElementById('divZei').style.visibility = "visible";
  // Buchstaben in der Grafik positionieren und anzeigen - Ende
  document.getElementById('antw').value = '';
  document.getElementById('antw').focus();
}
function nextPage() {
// nachfolgende Seite
  var j;
  num = num + 1;
  if (num > max-1) { num = max-1; return; }
  s = (num+1) + '/' + (max);
  document.getElementById('num').innerHTML = s;
  if (num == 0) { document.getElementById('myImg').style.width = 680 + 'px'; }
  else { document.getElementById('myImg').style.width = 580 + 'px'; }
  document.getElementById('divZei').style.visibility = "hidden";
  word = arr1[num];
  if (arr1[num] == '*') { j = Math.floor(anz1*Math.random()); word = arr4[j]; }
  bild = arr2[num];
  if (bild == 'no') { bild = 'no.jpg'; }
  snd = arr3[num];
  if (snd == '*') { snd = arr5[j]; }
  document.getElementById('buch').innerHTML = word;
  document.getElementById('myImg').src = bild;
  if (snd != 'no')  { soundplay(snd); }
  document.getElementById('antw').value = '';
  document.getElementById('antw').focus();
}
function backPage() {
// vorangehende Seite
  var i;
  num = num - 1;
  if (num < 0) { num = 0; return; }
  s = (num+1) + '/' + (max);
  document.getElementById('num').innerHTML = s;
  if (num == 0) { document.getElementById('myImg').style.width = 680 + 'px'; }
  else { document.getElementById('myImg').style.width = 580 + 'px'; }
  document.getElementById('divZei').style.visibility = "hidden";
  if (num == 0) { i = Math.floor(anz*Math.random()); word = letters.charAt(i); }
  else { word = arr1[num]; }
  bild = arr2[num];
  if (bild == 'no') { bild = 'no.jpg'; }
  snd = arr3[num];
  if (snd == '+') { help = word +'.mp3'; snd = help.toLowerCase(); }
  document.getElementById('buch').innerHTML = word;
  document.getElementById('myImg').src = bild;
  if (snd != 'no')  { soundplay(snd); }
  if (arr1[num] == '+') {
     x = getPosition('myImg').x + xKoord[i];
     y = getPosition('myImg').y + yKoord[i] + 8;
     setPosition('divZei',x,y);
     document.getElementById('zei').innerHTML = word;
     document.getElementById('divZei').style.visibility = "visible";
  }
  document.getElementById('antw').value = '';
  document.getElementById('antw').focus();
}
```

```
function repeatPage() {
// aktuelle Seite wiederholen
  var i,j;
  word = arr1[num];
  if (arr1[num] == '+') { i = Math.floor(anz*Math.random()); word = letters.charAt(i); }
  if (arr1[num] == '*') { j = Math.floor(anz1*Math.random()); word = arr4[j]; }
  if (num == 0) { document.getElementById('myImg').style.width = 680 + 'px'; }
  else { document.getElementById('myImg').style.width = 580 + 'px'; }
  bild = arr2[num];
  if (bild == 'no') { bild = 'no.jpg'; }
  snd  = arr3[num];
  if (snd == '+') { help = word +'.mp3'; snd = help.toLowerCase(); }
  if (snd == '*') { snd = arr5[j]; }
  if (snd != 'no')  { soundplay(snd); }
  document.getElementById('buch').innerHTML = word;
  document.getElementById('myImg').src = bild;
  if (arr1[num] == '+') {
     x = getPosition('myImg').x + xKoord[i];
     y = getPosition('myImg').y + yKoord[i] + 8;
     setPosition('divZei',x,y);
     document.getElementById('zei').innerHTML = word;
     document.getElementById('divZei').style.visibility = "visible";
  }
  document.getElementById('antw').value = '';
  document.getElementById('antw').focus();
}
</script>
</head>

<body onload = "initPage()" onkeyup="KeyListener(event)" oncontextmenu="return false">
<fieldset id="fs">
<h1>Sprache lernen</h1>
<span id="buch"> </span>
<br>
Schreibe und überprüfe mit [Test]
<br><br>
<input type="text" size="10" id="antw" class="cd1" autocomplete="off"/>
<br><br>
Zufallstext auf Seiten [1] und [8] mit [Repeat]<br>
(alle Bilder sind mit der Maus verschiebbar)
<br><span>------------------------------------------------------------</span><br>
Seite <span id = "num">1</span>  
<input type = "button" value=" Next " onclick="nextPage()" class="cd2"/>  
<input type = "button" value=" Back " onclick="backPage()" class="cd2"/>  
<input type = "button" value=" Repeat " onclick="repeatPage()" class="cd2"/>  
<input type = "button" value=" Test " onclick="pruef()" class="cd2"/>  
<br><span>------------------------------------------------------------</span><br>
<input type="button" value=" Mikro(+) " onclick="NewWindow()" class="cd2"/> 
<input type="button" value=" Mikro(-) " onclick="ShutWindow()" class="cd2"/> 
<input type="button" value="  Cam(+)  " onclick="NewWindow1()" class="cd2"/> 
<input type="button" value="  Cam(-)  " onclick="ShutWindow1()" class="cd2"/> 
<br><span>------------------------------------------------------------</span><br>
<audio id="splay" width="0"> <source src=" " type="audio/mp3"> </audio>
<div id="divImg">
   <img id="myImg">
</div>
<div id="divZei">
   <span id="zei"> </span>
<div>
<script>
   dragElement(document.getElementById("divImg"));
   document.getElementById('antw').focus();
</script>
</fieldset>
</body>
</html>
```

Hinweis: Im Programm kann mithilfe von Schaltern die Bildseite gewechselt werden. Zu jeder Seite gibt es entsprechende Bild- und Sounddateien, deren Namen am Programmanfang in Strings aufgelistet sind. Die Strings werden dann in Arrays umgewandelt. Dabei entspricht der Arrayindex der jeweiligen Seitennummer.

Sonderfälle sind die erste und die letzte Seite. Bei der ersten Seite wird, erkenntlich am Zeichen (+), mit einem Zufallsgenerator ein Buchstabe aus dem Array ausgewählt. Dieser wird vergrößert dargestellt und zusätzlich an der entsprechenden Position in der Tastatur-Grafik angezeigt. Außerdem wird noch die passende Sounddatei geladen. Bei der letzen Seite wird, erkenntlich am Zeichen (*), mit dem Zufallsgenerator aus dem Array ein Wort ausgewählt und dazu die passenden Bild- und Sounddateien geladen.

Anhang

HTML und CSS

Eine kurze Einführung

In dieser Einführung werden ausgewählte Elemente und Befehle aus der mächtigen Vielfalt von HTML und CSS dargestellt . . . . .

Teil A: HTML-CSS-Kurzanleitung

WWW-Internetseiten werden mittels **HTML** (HyperText Markup Language, einer eigenen Skript-Sprache) erstellt. Ein HTML-Dokument enthält neben dem eigentlichen Text Steuercodes (Tags), welche die unterschiedlichen Elemente einer Seite, wie Zeichen- und Absatzformate, Überschriften, eingefügte Grafiken oder Verweise (Links) definieren. Jene Elemente, die man hervorheben möchte, markiert man mit solchen Steuercodes. Entsprechende Programme, so genannte WEB-Browser, die ein HTML-Dokument geliefert bekommen, erkennen und interpretieren diese Steuerzeichen dann richtig.

Zur Erstellung von HTML-Seiten genügt ein einfacher Texteditor wie **NotePad** von Microsoft oder auch **HtmlEdit** vom Autor. Dabei werden alle Zeichen (auch die deutschen Umlaute und das scharfe ß) in einen erweiterten ASCII-Code kodiert, was durch die Anweisung <META charset="ISO-8559-1"> bewirkt wird. Verwendet man jedoch komplexere Editoren wie **NotePad++** oder **EditPadLite** werden entsprechend der Anweisung <META charset="UTF-8"> alle Zeichen im Unicode kodiert. Im Gegensatz zum ASCII-Code, der nur 1 Byte für jedes Zeichen besitzt, stehen jetzt für jedes Zeichen 2 Bytes zur Verfügung. Mit diesen 256*256 = 65536 Bytes können alle Sprachen der Welt kodiert werden. Um dabei die deutsche Sprache zu erhalten muss am Anfang das Tag <html lang = "DE"> geschrieben werden. Manche Editoren stellen dem eigentlichen Unicode noch drei Erkennungsbytes voran (BOM = Byte Order Mark).

HTML-Befehle (Tags) werden durch spitze Klammern markiert. Fast alle Befehle bestehen aus einem **einleitenden** und einem **abschließenden** Tag. Das einleitende Tag kann außer dem Befehl noch Attribute und Werte enthalten. Dem abschließenden Tag wird ein Slash (/) vorangestellt, es markiert das Ende des Befehls. Groß- und Kleinschreibung wird bei HTML-Tags nicht unterschieden und mehrfache Leerzeichen werden ignoriert. Der grundlegende Aufbau einer HTML-Seite sieht folgendermaßen aus:

```
<!DOCTYPE html>
<HTML> oder <HTML lang="DE">
<HEAD>
<TITLE> . . . . . </TITLE>                                  beschreibt den Seiteninhalt
<META charset="ISO-8859-1">                                 älterer Zeichensatz (mit Umlauten)
   oder
<META charset="UTF-8">                                      bestimmt den Zeichensatz „Unicode“
<META NAME="description" CONTENT="Beschreibung">            beschreibt den Seiteninhalt
<META NAME="author" CONTENT="Autor">                        nennt den Autor
<META NAME="keywords" CONTENT="Stichwörter">                bezeichnet Stichwörter
<META NAME="viewport" CONTENT="width=device-width, initial-scale=1.0">
                                                            responsive Screen-Anpassung
        nicht sichtbarer Dokumenten-Kopf
</HEAD>

<BODY>
        sichtbarer Dokumenten-Körper
</BODY>
</HTML>
```

Die META-Anweisungen im Dokumenten-Kopf dienen der allgemeinen Beschreibung der HTML-Seite. Der Inhalt des Dokumenten-Körpers kann aus Texten, Bildern, Sounds und Videos bestehen. Diese Elemente werden in ihrem Erscheinungsort (position, left, top), in ihrer Größe (size, width, height) und in ihrer Farbe (color) durch einschlägige Befehle bestimmt.

Unterstützt wird **HTML** dabei durch die besondere Auszeichnungsprache **CSS** (Cascading Style Sheets). Deren Befehle bestimmen die Art und Weise (style) der Erscheinung der HTML-Elemente. Die Angabe von Stilanweisungen kann im Dokumenten-Kopf zwischen den Tags **<style>** und **</style>** stehen.

Neben CSS kann noch **JavaScript** in eine HTML-Seite eingebunden werden. Das ist eine eigenständige Programmiersprache, welche auf die HTML-Objekte und deren Styles zugreifen und sie auch verändern kann. Die JavaScript-Befehle müssen immer zwischen den Tags **<script>** und **</script>** stehen.

Das Dokument-Objekt-Modell (DOM)

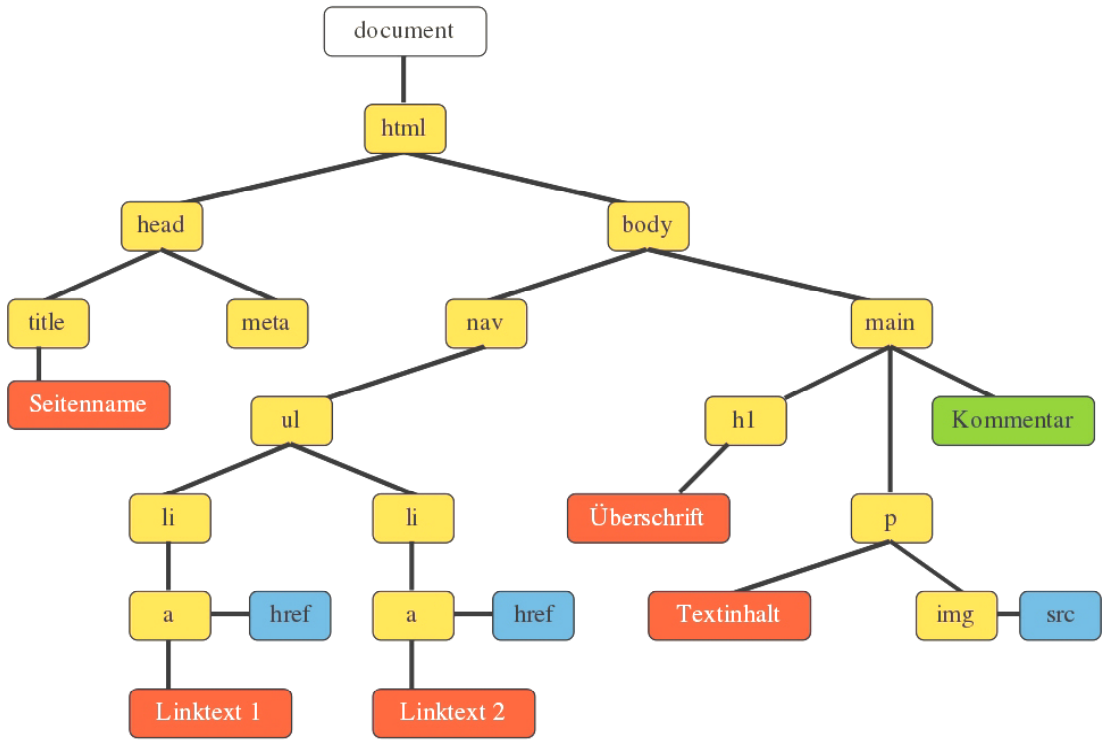
Ein Web-Browser auf dem Rechner des Benutzers (Client) stellt eine Verbindung mit dem Rechner des Anbieters (Server) über einen Internet-Link her und schickt an den Webserver eine Anfrage. Der Webserver antwortet mit einer Empfangsbestätigung und sendet an den Client ein HTML-Protokoll zurück.

Weil JavaScript im Web-Browser des Client integriert ist und auf dessen Rechner dann ausgeführt wird, ist JavaScript eine clientseitige Programmiersprache. Davon zu unterscheiden sind die serverseitigen Programmiersprachen (beispielsweise PHP), die nur am Server-Rechner laufen.

Der erhaltene HTML-Code liegt im Browser zunächst nur als Text vor. Während der Browser über das Netz den Code empfängt, wird durch einen so genannten Parser der Code schrittweise verarbeitet, d.h. in eine bestimmte Speicherstruktur im Arbeitsspeicher des Rechners übergeführt. Diese Speicherstruktur besteht aus zusammenhängenden Bündeln von Informationen, welche auch Objekte genannt werden. Die verschiedenen Objekte bilden eine Struktur, welche mit einem Wurzelknoten beginnt und sich dann baumartig zu weiteren Element-Knoten verzweigt. Der Aufbau dieser Objektstruktur ist durch das Document-Object-Model (DOM) geregelt. Das oberste Element (Wurzelknoten, root) trägt den Namen „***document***“, welches das HTML-Dokument im Browserfenster abbildet. Das „***document***“ enthält nun weitere Objekte: Zunächst „***HTML***“, „***HEAD***“, „***BODY***“, und dann beispielsweise Textfelder, Listen, Schaltflächen (buttons), Bilder (images), usw. Jedes Objekt besitzt bestimmte Eigenschaften (Attribute) und bestimmte Methoden (Funktionen). Die Attribute von übergeordneten Objekten werden auf untergeordnete Objekte vererbt.

An den einzelnen Objekten kann der Benutzer bestimmte Ereignisse (events) auslösen, beispielsweise mit der Maus auf einen Schalter klicken (click) oder auf der Tastatur eine Taste drücken (keypress).

Die Programmiersprache JavaScript kann nun auf die verschiedenen Objekte zugreifen und auch mögliche Ereignisse registrieren und mit bestimmten Unterprogrammen (Funktionen) darauf reagieren. Diese Behandlung von Ereignissen (event handling) ermöglicht erst die Interaktion von Benutzer und HTML-Dokument.



Schematische Darstellung eines DOM-Baumes mit HTML-Objekten.

Die wichtigsten HTML Elemente

Allgemeine Elemente	
<html> </html>	Erzeugt ein HTML Dokument
<head> </head>	Für Meta Informationen, z.B. Seitentitel.
<title> </title>	Name der Website. Wird im Browser oben angezeigt.
<body> </body>	Definiert den sichtbaren Teil der Website.
Formatierung	
	Für Absätze (Paragraphen). Nach einem Absatz wird automatisch ein Zeilenumbruch eingefügt.
	Definiert einen Abschnitt.
</ span>	Definiert einen Abschnitt in einer Zeile.
 	Zeilenumbruch.
<hr>	Fügt eine horizontale Linie ein.
Text	
 ... 	Fügt Überschriften ein. H1 ist die wichtigste (größte) Überschrift, H6 ist die kleinste Überschrift.
** **	**Bestimmt die Schriftart.**
** **	**Bestimmt die Größe des Textes.**
** **	**Bestimmt die Farbe des Textes.**
	Hebt Text **fett** hervor.
	Hebt Text **fett** hervor.
	Hebt Text *kursiv* hervor.
<u > </u >	Stellt Text unterstrichen dar.
Bilder	
<img>	Fügt ein Bild ein. Es muss immer das Attribut **src="..."** angegeben werden. Das Attribut **alt="..."** sollte angegeben werden. Die Größe kann über die Attribute **width="..."** und **height="..."** definiert werden.
Links	
<a> 	Fügt einen Link ein. Es muss immer das Attribut **href="..."** angegeben werden. Wenn ein Link in einem neuen Fenster geöffnet werden soll, kann man das Attribut **target="_blank"** nutzen.
Listen	
	Unsortierte Liste mit Markierungszeichen.
	Sortierte Liste mit Nummerierung.
	Linienelement zwischen Start-Tag und Ende-Tag einer Liste.

Tabellen

```
<table border="1">        Beginn einer Tabelle (nur zum Testen immer border="1" setzen)
 <thead>                  Bereich für den Kopf beginnt
  <tr>                    hier fängt die erste Reihe für die Kopfzeile an
    <th>Überschrift</th>  hier fängt die erste Zelle für die Kopfzeile (Überschriften für Spalte)
    <th>Ü. Spalte 2</th>  zweite Überschrift für zweite Spalte
 </tr>                    erste Reihe für Überschriften beenden
 </thead>                 Bereich für Kopf zu Ende

 <tfoot>                  Bereich für die Fußzeile (kommt immer vor Inhalt!)
  <tr>                    Beginn der Reihe für Fußzeile
   <td> Fuß Z1 </td>      Zell-Anfang, Inhalt Fußzeile 1, Zell-Ende
   <td> Fuß Z2 </td>      Zell-Anfang, Inhalt Fußzeile 2, Zell-Ende
  </tr>                   Ende  der Reihe für die Fußzeile
 </tfoot>                 Bereich für Fuß zu Ende

 <tbody>                  Bereich für Inhalt beginnt
  <tr>                    hier fängt die erste Reihe mit Inhalt an
    <td>Zelle 11 </td>    hier fängt die erste Zelle an, dann der eigentliche Inhalt und die Zelle beenden
    <td>Zelle 12 </td>    hier fängt die zweite Zelle an, dann der eigentliche Inhalt und die Zelle beenden
  </tr>                   erste Reihe mit Inhalt beenden

  <tr>                    nächste Reihe mit Inhalt
    <td>Zelle 21 </td>    erste Zelle, zweite Reihe
    <td>Zelle 22 </td>    zweite Zelle, zweite Reihe
  </tr>                   zweite Reihe mit Inhalt beenden

                          Hier können noch beliebig viele weiteren Reihen folgen

 </tbody>                 Bereich für Inhalt zu Ende

</table>                  Tabelle beenden
```

Formulare

Formulare werden für Eingabe, Ausgabe und Senden von Daten verwendet. Der Formularinhalt wird von den Tags **<form>** und **</form>** eingeschlossen. **<form action= "Zieladresse" method="get">** definiert, wohin die Daten gesendet werden, d.h. an Internetadressen (**url** = uniform resource locator). Wenn das Formular seine Daten an sich selbst (**self**) senden soll, sind neben **<form>** keine weiteren Attribute nötig. Der Formularinhalt kann aus verschiedenen Kind-Elementen bestehen:

(1) **<input type="text" name="…" value="…" size="…" maxlength="…">**
Einzeiliges Text-Feld mit Namen, Anfangswert, Länge und maximaler Länge.

(2) **<textarea name="…" cols="…" rows="…"> </textarea>**
Mehrzeiliges Text-Feld mit Namen, Spaltenanzahl und Zeilenanzahl.

(3) **<input type="submit" value="Senden">** Sendet das Formular an die angegebene Zieladresse.
<input type="button" value="Link" onClick = "location.href='Zieladresse'"> Sendet nach einem Mausklick das Formular an die angegebene Zieladresse.
<input type="reset" value="Reset"> Eine Schaltfläche zum Zurücksetzen des Formulars.

(4) **radio-, checkbox-, select- und option-**Elemente zur Auswahl von verschiedenen Möglichkeiten.

Formulare

<form action="" method=""> <!-- Hier kommen nun die gewünschten Formularelemente --> **</form>**		Zwischen diesen TAGs werden die einzelnen Formularfelder eingeschlossen **action=""** -> entweder Seite, die aufgerufen werden soll und der der Inhalt der Felder übergeben wird, oder Mailadresse in Form "mailto:donald@duck.ent" (nicht ratsam wegen SPAM!) **method="post|get"** – entweder post oder get
<input **type="text"** name="" value="" size="" maxlength="" />	Text-Box	**Text-Box** **name=""** -> Name des Feldes (für Variablenübergabe) **value=""** -> Vorgabewert (normalerweise leer) **size=""** -> Größe der Anzeige **maxlength=""** -> maximale Länge des Feldes
<input **type="hidden"** name="" value="" size="" maxlength="" />		versteckte Feld
<textarea name="" cols="10" rows="60"></textarea>	Text-Area	**Text-Area** Textblock für Eingabe von viel Text – **cols=""** -> wie viele Spalten angezeigt werden **rows=""** -> Anzeige der Anzahl Reihen
<input **type="radio"** name="" value="v" checked="checked" />	Radio-Button Frau | Herr	**Radio-Button** Wird gerne verwendet, wenn es um entweder-oder-auswahlen geht (z. B. Herr oder Frau) **name=""** -> Name des Feldes (für Variablenübergabe) pro Auswahl eine input-Zeile mit demselben Namen! value="v" -> Vorgabewert ist hier wichtig, da der Nutzer nur anklicken kann (Bsp: value="Herr")
<input **type="checkbox"** name="" checked="checked" />	Check-Box	**Check-Box** Anklickbox – über **checked="checked"** kann die Box bereits angeklickt sein
<select name="" size="3" multiple="multiple"> <option value="1" selected="selected">1</option> <option value="2">2</option> <option value="3">3</option> </select>	List-Box 1 2 3	**List-Box** Pull-Down Auswahl **size="3"** -> Größe der Anzeige **multiple="multiple"** -> ob mehrere Elemente angewählt warden können **<option ...** -> die einzelnen Elemente **selected="selected"** -> vorselektiertes
<input **type="Submit"** name="" value="speichern" />	Submit-Button speichern	Button zum Anklicken und Absenden des Formulars **value="speichern"** -> Der value wird als Beschriftung der Schaltfläche angezeigt

CSS-Befehle Funktion (Parameter)

Schrift

```
font-family          Schriftart  (Arial, Times New Roman, etc.)
font-size            Schriftgröße  (Prozent oder Pixel (px) )
color                Schriftfarbe  (red, green, blue usw., oder hexadezimal "#RRGGBB" )
font-variant         Schriftvariante  (normal, small-caps)
font-weight          Schriftgewicht  (normal, bold, bolder, lighter)
font-style           Schriftstil  (normal, oblique, italic)
```

Textgestaltung

```
text-align           Textausrichtung  (left, right, center, justify (Blocksatz))
line-height          Zeilenabstand  (Prozent oder Pixel (px) )
text-decoration      Textgestaltung  (underline, overline, line-through, blink)
word-spacing         Wortabstand  (Prozent oder Pixel (px) )
letter-spacing       Zeichenabstand  (Prozent oder Pixel (px) )
text-indent          Texteinrückung  (Prozent oder Pixel (px) )
text-transform       Textart  (capitalize, uppercase, lowercase, none)
```

Links

```
A:link               Farbangabe des Link-Tags ( color = …; )
A:visited            Besuchter Link ( color = …; )
A:hover              Link bei Mausberührung ( color = …; )
A:active             Angeklickter Link ( color = …; )
```

Bilder

```
background-color     Hintergrundfarbe  (red, green, blue usw., oder hexadezimal "#RRGGBB" )
background-image     Hintergrundbild URL("name.jpg")
background-attachement  Hintergrundbild fixiert (fixed) oder gescrollt (scroll)
background-repeat    Kacheln  (repeat, repeat-x, repeat-y, no-repeat)
```

Ränder

```
padding              Innenabstand allseitig  (Prozent oder Pixel (px) )
padding-top          Innenabstand oben  (Prozent oder Pixel (px))
padding-left         Innenabstand links   (Prozent oder Pixel (px))
padding-bottom       Innenabstand unten  (Prozent oder Pixel (px))
padding-right        Innenabstand rechts  (Prozent oder Pixel (px))

border               Dicke der Rahmenlinie allseitig  (thin, medium, thick oder Pixel (px))
border-top-width     Dicke der Rahmenlinie oben  (thin, medium, thick oder Pixel (px))
border-left-width    Dicke der Rahmenlinie links  (thin, medium, thick oder Pixel (px))
border-bottom-width  Dicke der Rahmenlinie unten  (thin, medium, thick oder Pixel (px))
border-right-width   Dicke der Rahmenlinie rechts  (thin, medium, thick oder Pixel (px))

border-style         Rahmentyp  (none,dotted,dashed,solid,double,groove,ridge,inset,outset)
border-color         Rahmenfarbe  (Farbname oder Hexadezimal)

margin               Außenabstand allseitig (Prozent oder Pixel (px) )
margin-top           Außenabstand oben (Prozent oder Pixel (px) )
margin-left          Außenabstand links (Prozent oder Pixel (px) )
margin-bottom        Außenabstand unten (Prozent oder Pixel (px) )
margin-right         Außenabstand rechts (Prozent oder Pixel (px) )

width                Rahmenbreite  (auto, Prozent, Pixel (px))
height               Rahmenhöhe  (auto, Prozent, Pixel (px))
```

LINKS

(1) **** Hinweisender Text ****

(2) **** Hinweisender Text ****

Bei Variante (2) wird das HTML-Dokument in einem neuen Fenster geöffnet.

Pseudoklassen

Folgende Formate werden verwendet, damit ein Feedback bei Links erreicht werden kann.

Beispiel mit CSS-Code:

Sehr wichtig ist die Reihenfolge (Faustformel dazu: LoVe HAte)

```
#navi a:link      { color:blue;     text-decoration:none; }
#navi a:visited   { color:black;    text-decoration:line-through; }
#navi a:focus     { color:green;    text-decoration:overline; }
#navi a:hover     { color:red;      text-decoration:overline; }
#navi a:active    { color:orange;   text-decoration:underline; }
```

Im HTML-Code dann

```
<div id="navi">
  <ul>
    <li><a href="index.htm">Startseite</a></li>
    <li><a href="ueber-mich.htm">Über mich</a></li>
    <li><a href="termine.htm">Termine</a></li>
    <li><a href="impressum.htm">Impressum</a></li>
  </ul>
</div>
```

link	Der normale Anzeige für einen Link
visited	Einen bereits vom Nutzer besuchter Link
focus	Den Zustand von **focus** erreicht man, wenn man über die TAB-Taste (öfters drücken) zu einem Link wandert. Dieser kann dann mit der Return gewählt werden.
hover	Mit der Maus über einen Link fahren
active	Gerade aktiver Link

Absolute/relative Positionierung

position	Art der Positionierung: relative | static | fixed | absolute;
position: absolute;	weiter notwendige Angaben left | right top | bottom width height
z-index	**Beispiel:** z-index: 1; Reihenfolge der Elemente (wird oft bei absolut positionierten Elementen benötigt) Je höher die Nummer, desto weiter im Vordergrund

Kurzschreibweisen

Für einige CSS-Formate gibt es die Möglichkeit, diese in einer Kurzschreibweise zusammenzufassen. Dadurch wird es übersichtlicher und weniger Code.

Kurzschreibweise in Beispielen

border:1px solid red;	Rahmen: border-width border-style border-color;
background: #ff00ee url(bild.jpg) fixed no-repeat right top;	background-color: #ff00ee; background-image: url(bild.jpg); background-attachment: fixed; background-repeat: no-repeat; background-position: right top;
padding: 35px 30px 20px 15px;	Leserichtung wie bei Uhr (oben, rechts, unten, links) Kurzschreibweise für: padding-top: 35px; padding-right: 30px; padding-bottom: 20px; padding-left: 15px;
padding: 35px 18px;	Wenn oben und unten gleich sind (wie auch rechts und links) Kurzschreibweise für: padding-top: 35px; padding-right: 18px; padding-bottom: 35px; padding-left: 18px;
padding: 8px;	Wenn alle 4 Seiten gleich sind Kurzschreibweise für: padding-top: 8px; padding-right: 8px; padding-bottom: 8px; padding-left: 8px;
margin:	Desselbe wie bei padding
font:	font-size und font-family sind Pflichtangaben, wobei die font-family immer die letzte Angabe ist. font: 12pt Arial, sans-serif; wenn auch die line-height mitgegeben wird, dann wird diese mit / von der Schriftgröße getrennt: font:12pt/1.6em serif;

CSS3

Responsives Webdesign: Das ist die automatische Anpassung der Größe der HTML-Objekte an den jeweiligen Geräte-Bildschirm (device-screen) von desktops, laptops, tablets und phones.

<META NAME="viewport" CONTENT="width=device-width, initial-scale=1.0">

Zusätzliche Media-Query – Anweisungen in CSS3:

@media (not | only) ***mediatype*** **and (*expression*) { CSS-Code }**

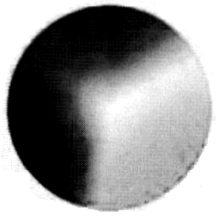
Beispiel:

```
@media screen and (min-width: 600px) and (max-width: 900px) {
    body { background-color: lightblue; font-size: 24px; border: 8px solid black; }
}
```

CSS3

Verschiedene CSS3-Anweisungen funktionieren nur in aktuellen Browserversionen!

Farbangaben CSS3

color: #00ff00;	alte Form der Farbangabe
color: rgb(0, 255, 0);	Farbangabe anhand RGB-Zahlen (dezimal)
color: rgb(0%, 100%, 0%);	Farbangabe in RGB anhand Prozenten
color: hsl(300, 100%, 60%);	Farbangabe im HSL-System H = hue = Farbton S = saturation = Sättigung L = lightness = Helligkeit H als Winkel von 0 – 360 S in Prozent von 0 – 100% L in Prozent von 0 – 100% 

CSS3: Transparenz

opacity:	0 ... 1 Grad der Durchsichtigkeit (vererbt die Eigenschaft!)
color: rgba(..., ..., ..., 0.5)	0 ... 1 | als 4 Wert zur Farbe als Transparenz
color: hsla(..., ..., ..., 0.5)	0 ... 1 | als 4 Wert zur Farbe als Transparenz

CSS3: Schatten

box-shadow: **X**px **Y**px Schattenverlauf Farbe;	Nun kann ein Schatten einem Element mitgegeben werden
box-shadow: **X**px **Y**px Schattenverlauf Farbe **,** **X**px **Y**px Schattenverlauf Farbe;	mehrere Schatten (Trennung erfolgt über Komma)
box-shadow: 10px 10px 100px gray; **-webkit-box-shadow:** 10px 10px 100px gray; **-moz-box-shadow:** 10px 10px 100px gray;	für verschiedene Browserhersteller!

CSS3: abgerundete Ecken

border-radius: **X**px , **Y**px;	Zum abrunden von Ecken – ist nur 1 Wert angegeben, gilt dieser für X und Y 
border-top-left-radius: border-top-right-radius: border-bottom-left-radius: border-bottom-right-radius:	Für jede Ecke einzeln definierbar
border-radius: 40px; -moz-border-radius: 40px; -webkit-border-radius: 40px; **Besonderheiten Firefox:** -moz-border-radius-topleft: 40px; -moz-border-radius-topright: 40px; -moz-border-radius-bottomleft: 40px; -moz-border-radius-bottomright: 40px;	für verschiedene Browser **Besonderheiten Webkit-Browser:** -webkit-border-radius: 40px 30px; /*Leerzeichen!*/

CSS3: Schatten bei Text

text-shadow: Xpx Ypx Schattenverlauf Farbe;	Beispiel: text-shadow: 5px 10px 8px orange;
text-shadow: -5px -5px 6px green, 5px 10px 6px red;	Mehrfarbiger Schatten

CSS3: Transformationen

transform: **rotate**(**X**deg);	**rotieren (drehen)** Beispiel: transform: rotate(**X**deg); WICHTIG: kein Leerzeichen nach rotate!! positiver Wert = Drehung im Uhrzeigersinn negativer Wert = Drehung gegen Uhrzeigersinn 
transform: rotate(30deg); -moz-transform: rotate(30deg); -ms-transform: rotate(30deg); -o-transform: rotate(30deg); -webkit-transform: rotate(30deg);	**rotate für verschiedene Browser**
transform-origin: X Y;	**Mittelpunkt für die Drehung festlegen** transform-origin: 0 0; /* obere linke Ecke */ transform-origin: 100% 0; /* obere rechte Ecke */ transform-origin: 100% 100%; /* untere rechte Ecke */ transform-origin: 50% 0; /* mitte oben */ 
transform: scale(wert); transform: scale(x, y);	**skalieren** bei Angabe eines Wertes wird x wie y vergrößert bei Unterschiedlicher Angabe von x wie y findet eine "Verzerrung" statt 
transform: scale(1.4); -moz-transform: scale(1.4); -ms-transform: scale(1.4); -o-transform: scale(1.4); -webkit-transform: scale(1.4);	**scale für verschiedene Browser**

transform: skew(wert);
transform: skew(x, y);
	schräg, (wind)schief, verdrehen
	Beispiel:
	transform: skew(20deg);
	transform: skew(20deg, 30deg);
	 

transform: skew(0, 20deg); -moz-transform: skew(0, 20deg); -ms-transform: skew(0, 20deg); -o-transform: skew(0, 20deg); -webkit-transform: skew(0, 20deg);	**skew für verschiedene Browser**

transform: translate(wert);
transform: translate(...px, ...px);
	CSS3 translate: umsetzen, verrücken, versetzen
	transform: translate(80px);
	transform: translate(80px, 50px);
	
	bei Angabe eines Wertes wird nur auf x versetzt angezeigt. Bei Angabe von x wie y findet in beiden Richtungen eine entsprechende versetzte Anzeige statt.

Browser-Spezifikationen: ***moz***: Mozilla-Firefox, ***ms***: Microsoft, ***o***: Opera, ***webkit***: Apple-Safari, …

CSS: Styles und Selektoren

(1) Globale Styles:

```
<!DOCTYPE html>
<head>
<style>

   Hier stehen die CSS-Definitionen im Dokumenten-Kopf

</style>
</head>
<body>
```

(2) Inline-Styles:

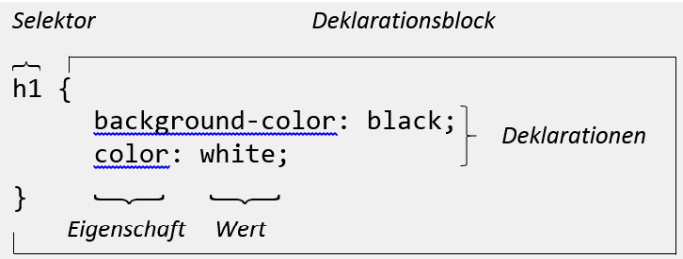
Hier stehen die CSS-Definitionen im Dokumenten-Körper.
Diese Form ist nicht optimal, weil so Inhalt (HTML) mit Präsentation (CSS) gemischt wird.
Ein Inline-Stylesheet wirkt sich nur auf ein Element aus. Beispiel:

```
<body>
  ...
  <img src="bild.jpg" width="240" height="160" alt="Beschreibung"
            style="border: 4px solid red">
 ...
</body>
```

(3) Externe Stylesheets:

Einbinden von externen Stylesheet-Dateien auf einer HTML-Seite im Dokumenten-Kopf.
Beispiel: **<link rel="stylesheet" href="myStyle.css"/>.**

Allgemeine Formatierungsregel für CSS-Befehle:



Beispiele:

```
h1 { font-size:20px; color:red; }

body {
   background-image: url("myPicture.jpg”);
   background-color: #D2B48C;
   margin-left: 20%;
   margin-right: 20%;
   padding: 10px 10px 10px 10px;
   font-family: "Times Roman", Arial;
}
```

Hinweis: Schriftnamen, die Leerzeichen enthalten, sollten in Anführungszeichen gesetzt werden!

Drei Arten der Selektoren-Bildung

(1) Durch Verweis auf einen Objekt-Namen [HTML-Element].
(2) Durch Verweis auf einen Identitäts-Bezeichner von Objekten [id].
(3) Durch Verweis auf einen Klassen-Bezeichner von Objekten [class].

Der einfachste Selektor besteht nur aus dem Namen des HTML-Elements, das formatiert werden soll, beispielsweise ein Absatz im Text (p):

```
p { color: red; }
```

Es kann auch ein Selektor für mehrere HTML-Elemente festgelegt werden:

```
p, h3 { color: red; }
```

HTML-Elemente können über ihre Identitätsbezeichnung **ID** oder über ihre **Klassen**-Bezeichnung angesprochen werden. Dazu werden die HTML-Tags um das Attribut **id="idname"** oder das Attribut **class="classname"** erweitert. Der Unterschied ist, dass mit **class** mehrere Elemente ausgezeichnet werden können, dagegen bezieht sich **id** immer nur auf ein bestimmtes Element pro HTML-Seite.

Für Klassen wird in der CSS-Definition vor dem Namen ein Punkt (**.**) geschrieben.
Für ID wird in der CSS-Definition vor dem Namen das Zeichen **#** geschrieben.

CSS-Definitionen im Dokumenten-Kopf:

```
#hinweis { font-style: bold; }
.bemerkung { font-style: italic; }
```

Entsprechende Objekte im Dokumenten-Körper:

```
<div id="hinweis">
     In der europäischen Union werden die Probleme größer.
     . . . . . .
</div>

<div class="bemerkung">
     In der europäischen Union werden die Probleme größer.
     . . . . . .
</div>
```

Es können auch mehrere Klassen-Selektoren für ein HTML-Element festgelegt werden:

```
p.eins { color: red; }
p.zwei { color: blue; }
```

Links können als Pseudo-Klassen - basierend auf ihrem jeweiligen Zustand – ausgezeichnet werden:

```
a:link { color: blue;  /* noch nicht besuchter Link */ }
a:visited { color: orange;  /* bereits besuchter Link */}
a:hover { color: green;  /* mit der Maus berührter Link */}
a:active { color: red; /* gerade besuchter Link */}
```

Vererbung

Viele CSS-Eigenschaften wirken nicht nur auf die vom Selektor ausgewählten HTML-Elemente, sondern werden auch an die „Nachkommen“ dieser Elemente vererbt. Wird beispielsweise im <**body**> die Schriftfarbe auf rot gesetzt, dann sind alle Textelemente, die sich im HTML-Körper befinden, ebenfalls rot.

Schriften (fonts)

font-family: Schriftart

Angabe der Schriften mit zusätzlicher, generischer Ersatzschrift (durch Beistriche getrennt).
Fonts, die Leerzeichen enthalten, werden in Anführungszeichen gesetzt.

Beispiel 1: font-family: Calibri, Arial, sans-serif; (Schriften ohne Serifen)
Beispiel 2: font-family: "Times New Roman", serif; (Schriften mit Serifen)
Beispiel 3: font-family: "Courier New", monospace; (Nicht proportionale Schriften)

font-size: Schriftgröße

px	**Pixel**
%	**Größe in Prozent in Relation zu einer anderen Schrift**
em	**Skalierungsfaktor für die Schriftgröße in Relation zu einer vorgegebenen Schriftgröße**

<**small**> kleingedruckter Text </**small**>

font-weight: Schriftstärke

Folgende Angaben sind möglich:
inherit (Schriftstärke des Elternelements)
lighter (dünner als im Elternelement)
normal (normale Schriftstärke)
bold (fette Schriftstärke)
bolder (fetter als im Elternelement)

font-style: Schriftstil

Folgende Angaben sind möglich:
inherit (Schriftstil des Elternelements)
normal (normaler Schriftstil, Voreinstellung)
italic (kursiver Schriftstil)
oblique (schräg gestellter Schriftstil)

line-height (Höhe der Textzeile)

Alle Schriftattribute können in einem einzigen Tag gesetzt werden.

Die Reihenfolge der Eigenschaften ist wichtig. Es müssen aber nicht alle Eigenschaften angegeben werden, nur **font-family** und **font-size** sind verpflichtend. Nicht angeführte Eigenschaften werden automatisch auf **normal** gesetzt. Die (optionale) Zeilenhöhe muss direkt nach der Schriftgröße und mit einem Schrägstrich angegeben werden.

font: [font-family] [font-style] [font-variant] [font-weight] [font-size]/[line-height];

Beispiel:
font: Arial, sans-serif italic small-caps bold 1.2em/1.6;

Beispiel:
<body> { font.size = 15px; }
<p {font-size: 1.2em; }> … </p>

In dem letzten Beispiel ergibt sich für den Absatz eine Schriftgröße von 15 * 1.2 = 18px.

Einschub über Zeichensätze

ASCII-Tabelle:

0		1		2		3		4		5		6		7	
8	BS	9		10	LF	11		12	FF	13	CR	14		15	
16		17		18		19		20		21		22		23	
24		25		26	EOF	27	ESC	28		29		30		31	
32		33	!	34	"	35	#	36	$	37	%	38	&	39	'
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
58	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	`	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127	
128		129	ü	130	é	131	â	132	ä	133	à	134	å	135	ç
136	ê	137	ë	138	è	139	ï	140	î	141	ì	142	Ä	143	Å
144	É	145	æ	146	Æ	147	ô	148	ö	149	ò	150	û	151	ù
152	ÿ	153	Ö	154	Ü	155	¢	156	£	157	¥	158	₧	159	ƒ
160	á	161	í	162	ó	163	ú	164	ñ	165	Ñ	166	ª	167	º
168	¿	169	⌐	170	¬	171	½	172	¼	173	¡	174	«	175	»
176	░	177	▒	178	▓	179	│	180	┤	181	╡	182	╢	183	╖
184	╕	185	╣	186	║	187	╗	188	╝	189	╜	190	╛	191	┐
192	└	193	┴	194	┬	195	├	196	─	197	┼	198	╞	199	╟
200	╚	201	╔	202	╩	203	╦	204	╠	205	═	206	╬	207	╧
208	╨	209	╤	210	╥	211	╙	212	╘	213	╒	214	╓	215	╫
216	╪	217	┘	218	┌	219	█	220	▄	221	▌	222	▐	223	▀
224	α	225	ß	226	Γ	227	π	228	Σ	229	σ	230	µ	231	τ
232	Φ	233	Θ	234	Ω	235	δ	236	∞	237	ø	238	ε	239	∩
240	≡	241	±	242	≥	243	≤	244	⌠	245	⌡	246	÷	247	≈
248	°	249	•	250	·	251	√	252	ⁿ	253	²	254	■	255	

Beispiel: Codierung des Strings „**Eva**“ mithilfe der ASCII-Tabelle im 10er-, 16er- und 2er-System.

E = 69, v = 118, a = 97

$69 = 4 * 16^1 + 5 * 16^0 = 45h = \underset{4h}{0100}\ \underset{5h}{0101} = 1 * 2^6 + 1 * 2^2 + 1 * 2^0$

$118 = 7 * 16^1 + 6 * 16^0 = 64h = \underset{7h}{0111}\ \underset{6h}{0110} = 1 * 2^6 + 1 * 2^5 + 1 * 2^4 + 1 * 2^2 + 1 * 2^1$

$97 = 6 * 16^1 + 1 * 16^0 = 61h = \underset{6h}{0110}\ \underset{1h}{0001} = 1 * 2^6 + 1 * 2^5 + 1 * 2^0$

E = 69 = 45h = 01000101
v = 118 = 76h = 01110110
a = 97 = 61h = 01100001

Drückt man auf eine Taste der Tastatur, dann erkennt der Computer den Bytewert des Zeichens. Eine intern gespeicherte Codetabelle (ASCII, American Standard Code for Information Interchange) wandelt bei der Bildschirm- oder Druckerausgabe diesen Code in das grafische Erscheinungsbild des Zeichens um. Im westeuropäischen Sprachraum verbreitete sich die Codetabelle ISO-8850-1, die eine modifizierte ASCII-Tabelle ist und u.a. auch die deutschen Umlaute enthält. Der Nachteil ist die 1-Byte-Darstellung, die nur 256 verschiedene Zeichencodes erlaubt. Dieser Nachteil wurde durch die Einführung des 2-Byte-Unicodes UFT-8 behoben, der jetzt 256*256 = 65536 Zeichen darstellen kann (genug für alle Sprachen der Welt). In HTML gibt es zusätzlich für ganz bestimmte Zeichen so genannte Sondercodes (auch Escape-Sequenzen genannt), beispielsweise:

Ä	`&Auml;`	Ü	`&Üuml;`	©	`&copy;`	¶	`&pi;`
ä	`&auml;`	ü	`&uuml;`	"	`&quot;`	∫	`&int;`
Ö	`&Ouml;`	ß	`&szlig;`	<	`<`	Σ	`&sum;`
ö	`&ouml;`	€	`&euro;`	>	`>`	space	` `

`<SUB>, </SUB>`	Beginn und Ende von tiefgestelltem Text.
`<SUP>, </SUP>`	Beginn und Ende von hochgestelltem Text.

Der Farbwert eines Zeichens kann auch hexadezimal codiert werden mit **#RRGGBB**, wobei jeweils 2 Bytes für die Grundfarben rot (R), grün (G) und blau (B) zur Verfügung stehen.

HTML-Elemente positionieren

Elemente können durch Verwendung der Eigenschaft **position** aus dem normalen Elementfluss entfernt werden und an eine beliebige andere Stelle gesetzt werden.

Mit **position: absolute** kann man Elemente losgelöst vom Textfluss positionieren, an eine Stelle, die mit meistens **top** und **left** festgelegt wird. Auch Größenangaben oder Abstände, wie **width** und **height** oder wie **margin** und **padding**, sind möglich.

Mit **position: relative** wird die Position nur um die angegebenen Werte (+ oder -) verschoben.

```
h1 {
   position: absolute;
   top: 40px;
   left: 100px;
}
```

float setzt Elemente an die linke oder rechte Seite eines Blocks, währen der restliche Text um das Element herumfließt. Mit **float** können die Elemente nebeneinander angeordnet werden.

```
img { float: left; }
```

Wenn Textabsätze mit links oder rechts schwebenden Bildern nicht so hoch sind wie das Bild, entstehen in aufeinander folgenden Textblöcken Treppenstufen. Um Treppen beim Umfließen von Bildern zu vermeiden, muss mit **clear** das Umfließen beendet werden, so dass die Elemente wieder untereinander stehen.

```
img { float: left; }
h2  { clear: left; }
```

Im normalen Elementfluss sind die Elemente (bzw. ihre Boxen) nicht positioniert oder gefloatet. Mit der Eigenschaft **display** wird festgelegt, in welcher Art von Box ein Element erscheint. Elemente mit **display: inline;** erzeugen eine oder mehrere **Inline-Boxen**. Inline-Boxen verlaufen auf einer Zeile horizontal in der Schreibrichtung der verwendeten Sprache. Inline-Boxen können Innen- und Außenabstände sowie Rahmen besitzen.

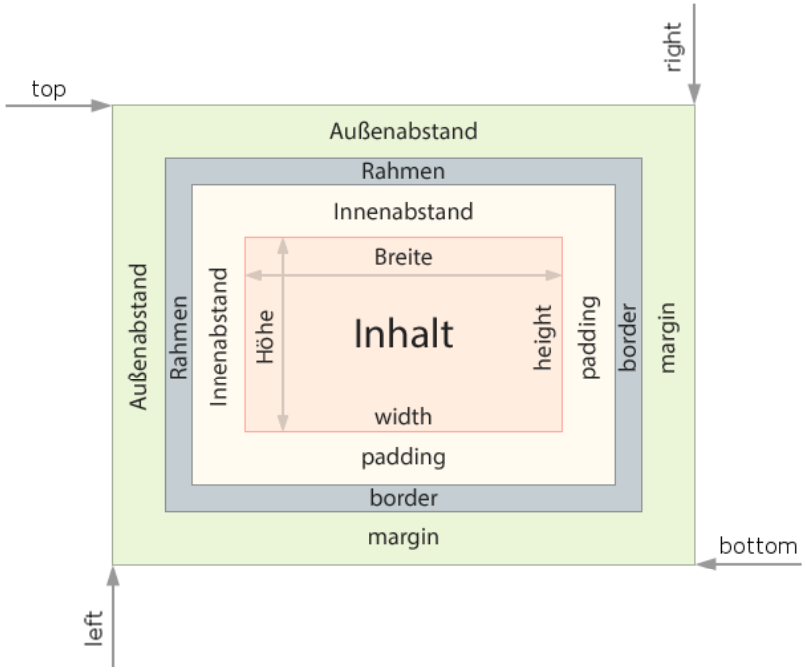
Block-Boxen nehmen die gesamte Breite des Elternelementes ein. Sie sind so hoch, wie ihr Inhalt. Dadurch entsteht ein zusammenhängendes Rechteck, das aussieht wie eine Box, und das genau hat dieser Darstellungsart ihren Namen gegeben.

Das klassische Box-Modell

CSS behandelt eine Webseite so, als wäre jedes darin enthaltene Element in einer unsichtbaren Box eingeschlossen. Eine Box besteht aus einem Inhalt (content), dem diesen Inhalt umgebenden Innenraum (**padding**), der Außenkante des Paddings (Rahmen, **border**) und dem unsichtbaren Bereich um diesen Rahmen (Rand, **margin**), der die Elemente voneinander trennt.

Außenabstand: margin-top, margin-right, margin-bottom, margin-left
Innenabstand: padding-top, padding-right, padding-bottom, padding-left

Beispiele:	`margin-left: 5px;`
	`padding-top: 10px;`
	`margin: 10px 5px 10px 5px;`
	`padding: 10px 5px 10px 5px;`



Durch die Angabe von **margin** oder **padding** ohne Zusatz von **top, right, bottom** oder **left** können die Abstände für alle vier Positionen festgelegt werden

Eine Angabe: für alle vier Abstände gilt derselbe Wert.
Zwei Angaben: 1. Wert für top und bottom. 2. Wert für right und left.
Drei Angaben: 1. Wert für top, 2. Wert für right und left. 3. Wert für bottom.
Vier Angaben: 1. Wert für top. 2. Wert für right. 3. Wert für bottom. 4. Wert für left.
(Merkhilfe: Uhrzeigerbewegung)

Treffen zwei untereinanderliegende Ränder (margins) aufeinander (collapsing), dann verschmelzen sie so, dass nur der größere Rand (margin) dargestellt wird.

Maßangaben:

X px = X Bildpunkte (Pixel)
X % = X Prozent relativ zum umgebenden Element
X em = X mal so groß wie das jeweilige Element

Mit **margin: auto** können Block-Elemente zentriert werden. Dazu muss aber eine feste Breite des Eltern-Elements definiert sein. Mit **text-align: center** können definierte Bereiche der Seite zentriert werden.

Über die Eigenschaft **border** kann die Rahmendicke, der Rahmentyp und die Rahmenfarbe festgelegt werden. Es ist auch möglich, Angaben für Rahmendicke, Rahmenfarbe und Rahmentyp für einzelne Seiten des Elements zu machen.

border-top definiert einen Wert für oben,
border-right für rechts,
border-bottom für unten,
border-left für links.

Beispiele:	`border: 1px solid red;`
	`border-bottom: thick double blue;`

Spezielle Strukturelemente von HTML5

Folgende Elemente steuern den grundsätzlichen Aufbau (layout) einer Internetseite mit HTML5, d.h. die Seitenstrukturierung:

body	**gesamter Inhaltsbereich**
header	**einleitende Inhalte**
nav	**Seitennavigation**
main	**Hauptinhalt**
section	**thematische Gruppierung von Inhalten**
article	**einzelne in sich geschlossene Inhaltsbereiche**
aside	**ergänzende Hinweise, Randbemerkungen**
footer	**informative Inhalte etwa Impressum, Copyright, Autor**
address	**Kontaktinformationen**

Der **header** enthält den sichtbaren Kopfbereich einer Webseite oder eines Teils einer Seite, der für einleitende Inhalte gedacht ist (Firmenlogos, Links zum Impressum oder zur Kontaktseite, …).

```
<header>
   <img src="/images/logo.png" alt="Firmenlogo">
   <h1>Herzlich Willkommen!</h1>
</header>
```

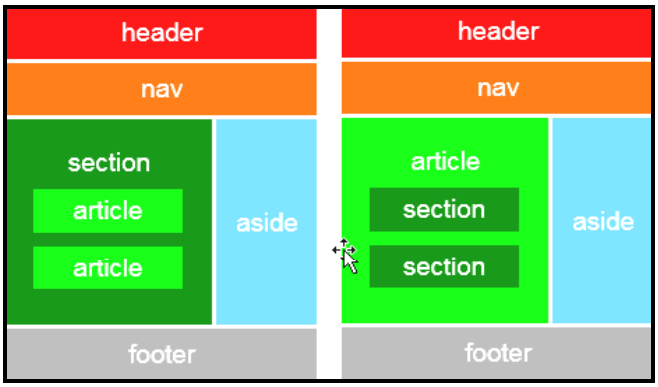
Das **nav**-Element umschließt Navigationsleisten und Menüs, wobei es neben einer unsortierten Liste mit den entsprechenden Links auch eine Überschrift oder ähnliches enthalten kann.

```
<nav>
   <h2>Navigation</h2>
      <ul>
         <li><a...> Link 1</a></li>
         <li><a...> Link 2</a></li>
         <li><a...> Link 3</a></li>
      </ul>
</nav>
```

Das **section**-Element dient zur Unterteilung von Inhalten nach inhaltlichen Gesichtspunkten. Es enthält zumeist eine thematische Gruppierung von Inhalten – sehr oft mit einer Überschrift.

Das **article**-Element stellt einen in sich geschlossenen Abschnitt eines Dokuments dar, vergleichbar mit einem Zeitungsartikel. Der Unterschied zum **section**-Element ist der, dass die Inhalte für sich alleine stehen. Es sind sozusagen isolierte Beiträge wie Blog-Postings oder Webseiten-Inhalte. Dies bedeutet, dass ein oder mehrere **article**-Elemente in einem themenbezogenen **section**-Element optimal platziert werden können. Das **section**-Element sollte verwendet werden, wenn es auf einer Seite tatsächlich verschiedene Themenbereiche gibt, die sich zusammenfassen lassen.

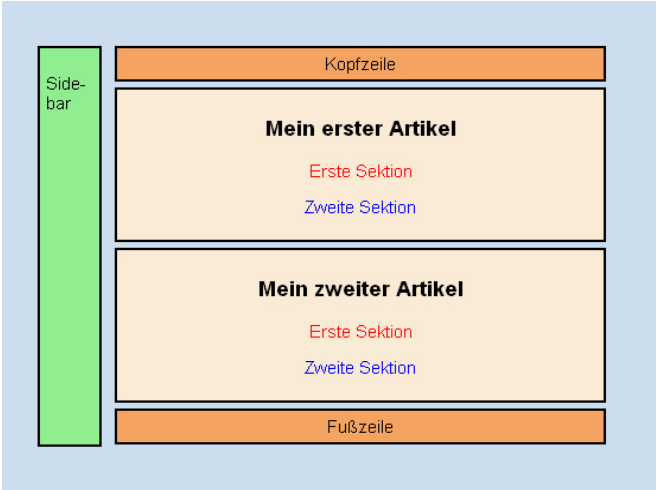
Es kann aber auch in **articles** thematische Gruppierungen von Inhalten mit einer Überschrift geben. Für solche Fälle sind **section**-Elemente innerhalb eines **article**-Elements die richtige Wahl. Die nachfolgende Grafik veranschaulicht diese beiden Varianten einer Seitenstrukturierung.



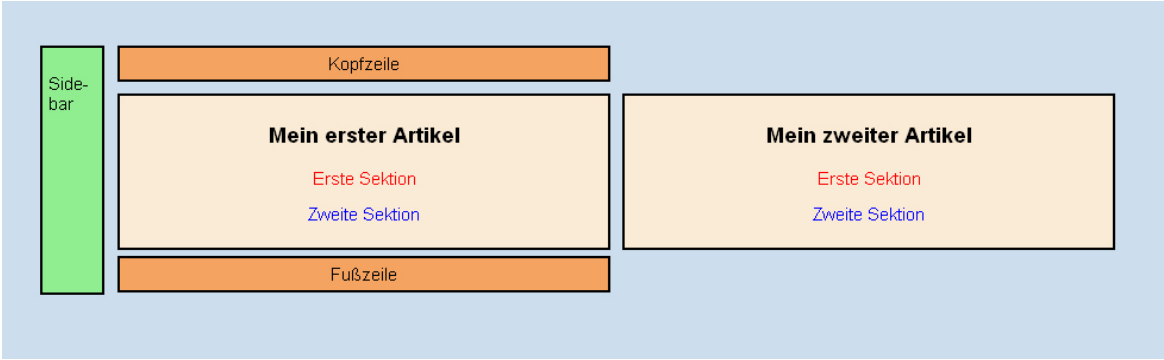
Das **aside**-Element umschließt laut Spezifikation Abschnitte einer Seite, deren Inhalte nur in einem indirekten Zusammenhang mit dem umgebenden Inhalt stehen. Dies sind zum Beispiel Randbemerkungen, Fußnoten oder Links zu weitergehenden Webseiten. Die Platzierung des **aside**-Elementes neben dem Hauptinhalt erfolgt über CSS.

Das **footer**-Element enthält alle Informationen, die in Webseiten am Ende stehen: Autor, Hinweise zum Urheberrecht, ein Link zum Impressum. Dabei kann ein **footer** letztes Element der Seite, aber auch das Ende eines **article**-Elementes sein.

Die folgende Grafik ist ein Snapshot des Programms „**boxen.html**“, das auf der nächsten Seite aufgelistet ist. Die Position des Sidebars ergibt sich aus den Maßangaben der restlichen HTML-Elemente.



Man erhält die folgende Grafik, wenn man im Programm „**boxen.html**“ die Kommentarklammern /* und */ entfernt. Dadurch werden die beiden Artikel horizontal gefloatet.



```
<!DOCTYPE html>
<html>
<head>
<title> Boxen  </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" Boxen ">

<style>
body{ background-attachment: fixed; font-family: arial; font-size: 15px;
      background-color: #CCDDEE; color: black; margin-left: 100px; margin-top: 50px;}
section.sec1 { color: red; }
section.sec2 { color: blue; }
.headfoot {
   /* clear: left; */
   width: 400px;
   padding: 5px;
   border: 2px solid black;
   background-color: sandybrown;
   margin: 5px;
   text-align: center;
}
.art {
   /* float: left; */
   width: 400px;
   padding: 5px;
   border: 2px solid black;
   background-color: antiquewhite;
   margin: 5px;
   text-align: center;
}
.sidebar {
   position: absolute; top:50px; left:40px;
   width: 40px;
   height: 335px;
   /* height: 200px; */
   background-color: lightgreen;
   padding: 5px;
   border: 2px solid black;
   text-align: left;
}
</style>
</head>

<body>
  <header class = "headfoot">
    Kopfzeile
  </header>
  <article class = "art">
    <h3> Mein erster Artikel </h3>
    <section class = "sec1">
      <p> Erste Sektion </p>
    </section>
    <section class = "sec2">
      <p> Zweite Sektion </p>
    </section>
  </article>
  <article class = "art">
    <h3> Mein zweiter Artikel </h3>
    <section class = "sec1">
      <p> Erste Sektion </p>
    </section>
    <section class = "sec2">
      <p> Zweite Sektion </p>
    </section>
  </article>
  <footer class = "headfoot">
    Fußzeile
  </footer>
  <aside class = "sidebar">
    <br>
    Side-<br>
    bar<br>
    <br>
  </aside>
</body>
</html>
```

Teil B: Verschiedene DEMO-Programme

Als geschichtliche Anmerkung sei noch erwähnt, dass im Jahr 2013 die HTML-Version **HTML 5** im Internet eingeführt wurde, vorher war **HTML 4** der Standard. HTML 5 bietet im Vergleich zu HTML 4 eine größere und funktionsmächtigere Vielfalt an Befehlen. Jedoch ist HTML 5 abwärts kompatibel, so dass auch ältere Befehle aus HTML 4 verstanden werden, beispielsweise der **** Befehl. Damit kann jener Text zwischen **** und **** entsprechend dem Attribut (*color*, *size*) ausgezeichnet werden. In HTML 5 wird diese Textauszeichnung mit entsprechenden CSS-Befehlen erreicht. Ein anderer alter Befehl aus HTML 4 ist der ** ** Befehl. Damit werden zwischen ** ** und ** ** liegende Elemente zentriert. Auch das Zentrieren kann in HTML 5 mit CSS-Befehlen ausgeführt werden.

Zur ersten einführenden Demonstration seien drei Musterseiten programmiert (*demo1.html*, *demo2.html* und *demo3.html*). Die erste Seite enthält einige HTML-Objekte und die Verweise (*links*) auf die beiden anderen Seiten. Die zweite Seite enthält in einer Tabelle (*table*) einen Monatskalender. Die dritte Seite enthält einen einfachen Rechner, der mit nur wenigen JavaScript-Befehlen realisiert ist. Der Leser möge die einzelnen Befehle in der Kurzanleitung nachlesen und dann analysieren, vor allem den Syntax der globalen Style-Befehle im Dokumenten-Kopf. Sie verweisen immer auf ein bestimmtes HTML-Objekt und ordnen diesem dann spezielle Attribute zu.

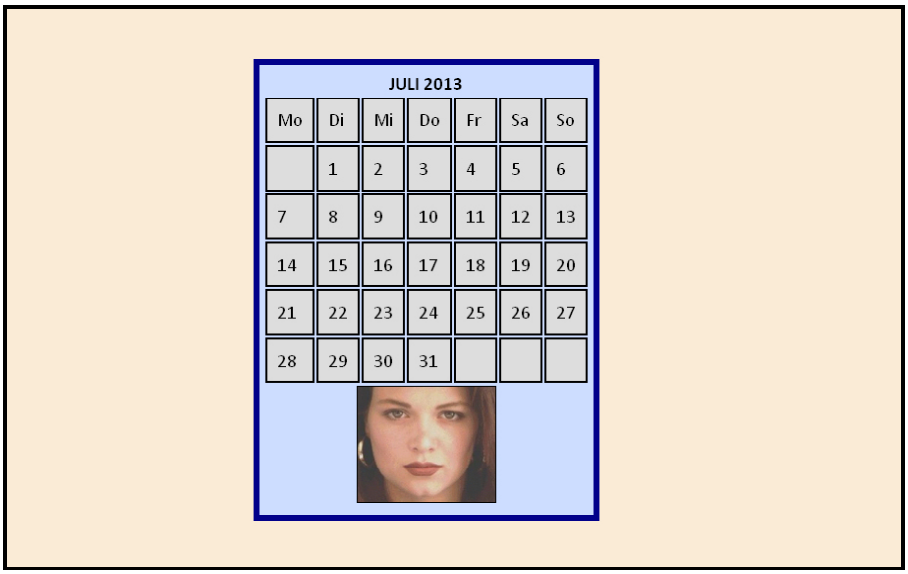
Insgesamt werden im Folgenden zehn einfache HTML-Programme (*demo1.html* bis *demo10.html*) aufgelistet. Drei Programme enthalten einen kurzen JavaScript-Code zwischen den Tags <script> und </script>.

(1) demo1.html – Schriften, Bilder und Links

```
<!DOCTYPE html>
<html>
<head>
<title> Erste Demoseite  </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" Erste Demoseite ">

<style>
body {
  background-image: url("back.jpg"); background-color:#CCDDEE; color:black;
  font-family:Calibri,Arial; font-size:19px; font-weight:normal;
  margin: 50px; padding-left:50px;
}
a:link {color: blue;}
a:visited {color: blue;}
a:hover {color: yellow;}
a:active {color: yellow;}
</style>

<body>
<a name="top"></a>
<a href="#bottom">Springe zum Seitenende "BOTTOM"</a>
<br><br>
Normale Textzeile<br>
<p style="color:red; font-weight:bold;font-style:italic;">Rote, fette, schräge Textzeile</p>
<span style="font-family:Arial">Textzeile in ARIAL</span><br>
<span style="font-family:Times Roman">Textzeile in TIMES ROMAN</span><br>
<span style="font-family:Courier New">Textzeile in COURIER NEW</span><br>
<br>
<a href="frau.jpg" target="_blank"><img src="frau.jpg" height="120px" border="1">
  Klick auf das Bild!
</a>
<br><br>
<a href="demo2.html">Springe zur externen Seite "demo2.html"</a>
<br><br>
<a href="demo3.html">Springe zur externen Seite "demo3.html"</a>
<br><br>
<span style="font-size: 19px">Textzeile in Schriftgröße "normal"</span><br>
<span style="font-size: 0.75em">Textzeile in Schriftgröße "0.75%"</span><br>
<span style="font-size: 1.25em">Textzeile in Schriftgröße "125%"</span><br>
<span style="font-size: 1.50em">Textzeile in Schriftgröße "150%"</span><br>
<span style="font-size: 1.75em">Textzeile in Schriftgröße "175%"</span><br>
<span style="font-size: 2.00em">Textzeile in Schriftgröße "200%"</span><br>
<span style="font-size: 19px">Textzeile in Schriftgröße "normal"</span><br>
<br>
<a href="#top">Springe zum internen Seitenanfang "TOP"</a>
<br>
<a name="bottom"></a>
</body>
</html>
```

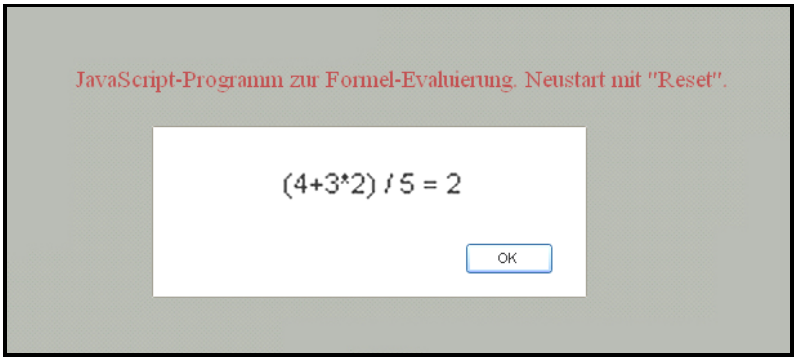
Hinweise:
Das Programm enthält für verschiedene Textzeilen **inline-Styles** zur Textauszeichnung.
Die Schriftgröße bezieht sich mittels **%** **em** auf die normale Schriftgröße.
Die HTML-Tags <**span**> Textzeile . . . </**span**> dienen zur einfachen Textmarkierung.

(2) demo2.html – Beispiel einer Tabelle



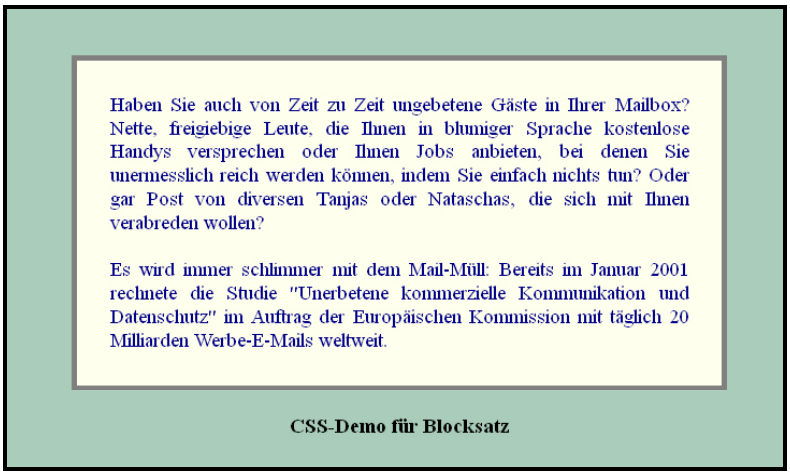

```
<!DOCTYPE html>
<html>
<head>
<title> Zweite Demoseite  </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" Zweite Demoseite ">

<style>
   body  {
         background-color:antiquewhite; color:black;
         font-family:Calibri,Arial; font-size:19px; font-weight:normal;
         margin: 25px; padding-left:25px;
   }
   table { background-color:#DDEEFF; border: 6px solid darkblue; padding:5px; margin: auto; }
   td { background-color:#DDDDDD; border: 2px solid black; padding:10px; }
</style>

</head>

<body>
<br>
<br>
<table>
  <th colspan=7>JULI 2013</th>
  <tr>
     <td>Mo</td> <td>Di</td> <td>Mi</td> <td>Do</td> <td>Fr</td> <td>Sa</td> <td>So</td>
  </tr>
  <tr>
     <td></td> <td>1</td> <td>2</td> <td>3</td> <td>4</td> <td>5</td> <td>6</td>
  </tr>
  <tr>
     <td>7</td> <td>8</td> <td>9</td> <td>10</td> <td>11</td> <td>12</td> <td>13</td>
  </tr>
  <tr>
     <td>14</td> <td>15</td> <td>16</td> <td>17</td> <td>18</td> <td>19</td> <td>20</td>
  </tr>
  <tr>
     <td>21</td> <td>22</td> <td>23</td> <td>24</td> <td>25</td> <td>26</td> <td>27</td>
  </tr>
  <tr>
     <td>28</td> <td>29</td> <td>30</td> <td>31</td> <td></td> <td></td> <td></td>
  </tr>
  <th colspan=7> <img src="frau.jpg" width="140" border="1"> </th>
</table>
</body>
</html>
```

(3) demo3.html – Einfache Formelauswertung mit JavaScript




```
<!DOCTYPE html>
<html>
<head>
<title> Dritte Demoseite  </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" Dritte Demoseite ">

<style>
    body {background-color: #E8F0D8;}
    p {color: red; font-size: 120%;}
</style>

</head>

<body>
   <p>JavaScript-Programm zur Formel-Evaluierung. Neustart mit "Reset".</p>

   <script>
      formel = prompt('Bitte eine Formel eingeben');
      ergebnis = eval(formel);
      alert(formel + ' = ' + ergebnis);
   </script>

</body>
</html>
```

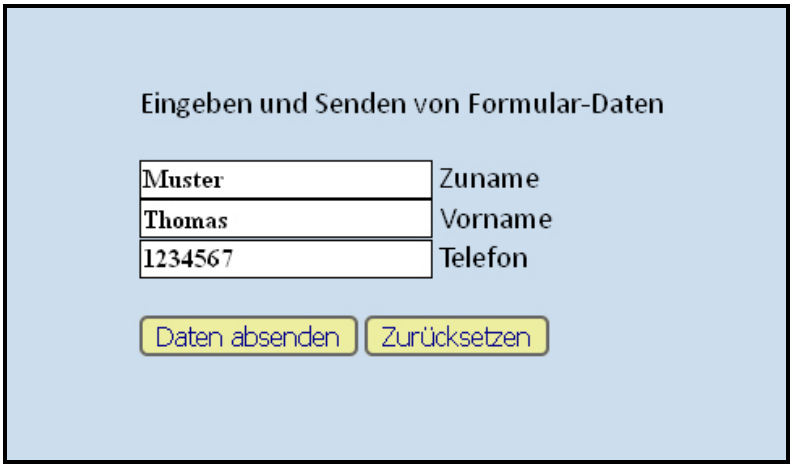
(4) demo4.html – Schriftauszeichnung mit Blocksatz




```
<!DOCTYPE html>
<html>
<head>
<title> CSS-Blocksatz </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Herbert Paukert">
<meta name="description" content=" CSS-Blocksatz ">
```

```
<style>
   body { background-color: #AACCBB;
   }
   #block { background-color:#FFFFEE; color:darkblue; width:600px; padding:5%;
            border-width:5px; border-style:solid; border-color:#808080;
            font-family:"Times Roman"; font-size:14pt; font-style:normal;
            text-align:justify;
   }
</style>
</head>

<body>
<br><br><br><br><br>
<table style="margin:auto;">
   <tr>
      <td id="block">
        Haben Sie auch von Zeit zu Zeit ungebetene Gäste in Ihrer Mailbox?
        Nette, freigiebige Leute, die Ihnen in blumiger Sprache kostenlose Handys
        versprechen oder Ihnen Jobs anbieten, bei denen Sie unermesslich reich
        werden können, indem Sie einfach nichts tun? Oder gar Post von diversen
        Tanjas oder Nataschas, die sich mit Ihnen verabreden wollen?
        <br><br>
        Es wird immer schlimmer mit dem Mail-Müll: Bereits im Januar 2001 rechnete
        die Studie "Unerbetene kommerzielle Kommunikation und Datenschutz" im Auftrag
        der Europäischen Kommission mit täglich 20 Milliarden Werbe-E-Mails
        weltweit.
      </td>
   </tr>
   <th> <h3>CSS-Demo für Blocksatz</h3> </th>
</table>
</body>
</html>
```

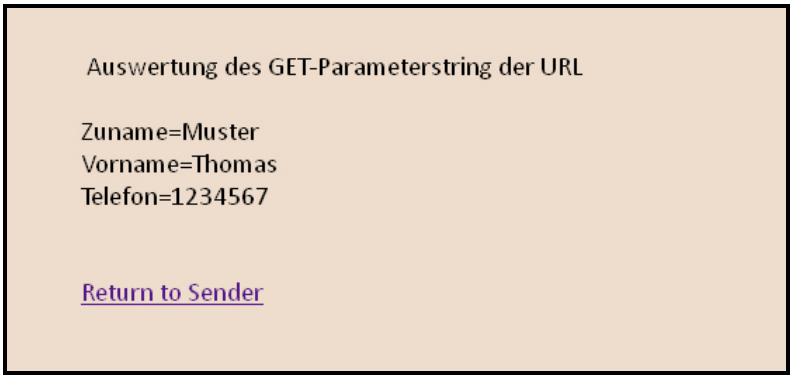
(5) demo5.html – Formulare senden




```
<!DOCTYPE html>
<html>
<head>
   <title> Formular senden </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <meta name="description" content=" Formular senden ">
   <style>
      body {
        background-color:#CCDDEE; color:black;
        font-family:Calibri; font-size:19px; font-weight:normal;
        margin:50px; padding-left:50px;
      }
     .ein {font-family:"Times Roman"; font-size:16px; font-weight:bold;
           border: 1px solid #000000;
      }
     .btn {background-color:#EEEEA0; color:darkblue; font-size:16px;
           border:2px solid #666666; border-radius:5px;
      }
   </style>
</head>
```

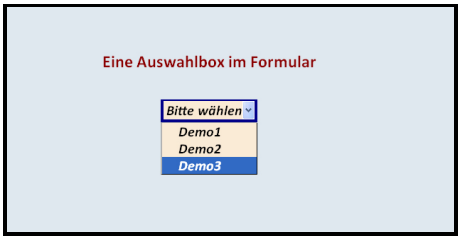
```
<body>
  <form action="demo6.html" method="get" enctype="text/plain">
    <br>Eingeben und Senden von Formular-Daten<br><br>
    <input class="ein" type="text" name="Zuname" value="Muster" size="20"> Zuname<br>
    <input class="ein" type="text" name="Vorname" value="Thomas" size="20"> Vorname<br>
    <input class="ein" type="text" name="Telefon" value="1234567" size="20"> Telefon<br>
    <br>
    <input class="btn" type="submit">
    <input class="btn" type="reset">
  </form>
</body>
</html>
```

(6) demo6.html – Formulare empfangen



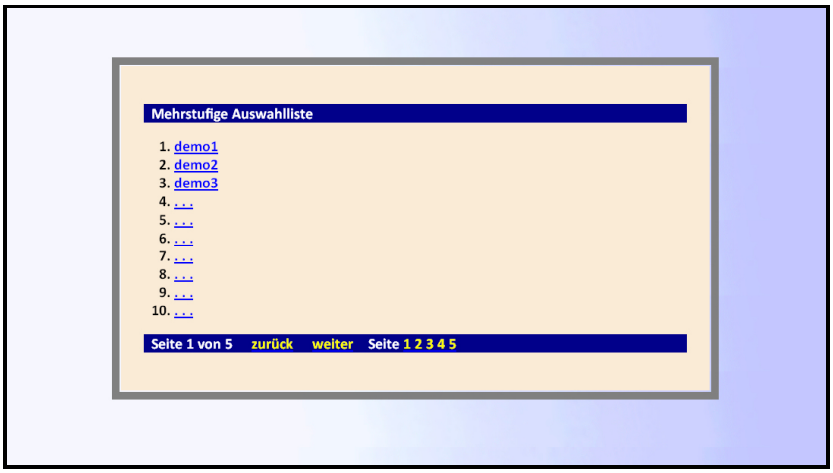

```
<!DOCTYPE html>
<html>
<head>
   <title> Formular empfangen </title>
   <meta charset="ISO-8859-1">
   <meta name="author" content="Herbert Paukert">
   <meta name="description" content=" Formular empfangen ">

   <style>
      body {
        background-color:#EEDDCC; color:black;
        font-family:Calibri; font-size:19px; font-weight:normal;
        margin:50px; padding-left:50px;
      }
    </style>

</head>

<body>
<br>
   Auswertung des GET-Parameterstring der URL
<br><br>

<script>
  with (document)
  {
    x = location.search.substring(1);
    a = x.split("&");
    for (var i = 0; i < a.length; i++)
       { write("  " + a[i] + "<br>"); }
  }
</script>

<br><br>
  <a href="demo5.html">Return to Sender</a>
<br>
</body>
</html>
```

(7) demo7.html – Beispiel einer Auswahlbox



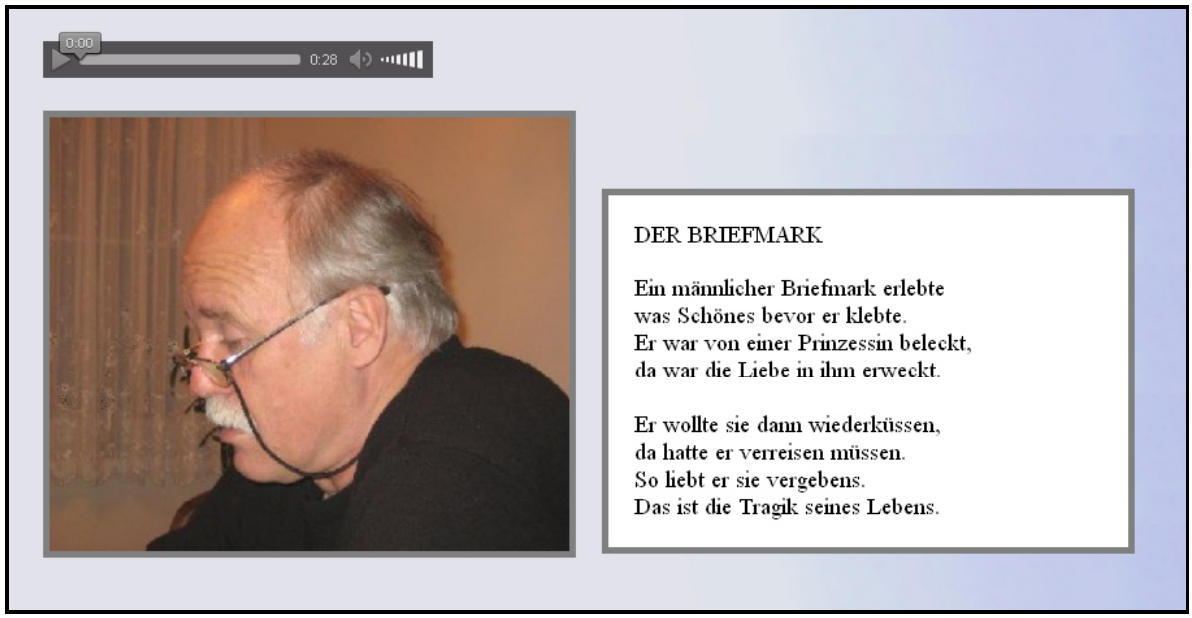

```
<!DOCTYPE html>
<html>
<head>
   <title>Auswahlboxen</title>
   <meta charset="ISO-8859-1">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta name="author" content="Herbert Paukert">
   <meta name="description" content="Auswahlboxen">
   <style type="text/css">
       body  {background-color:#E0E8EF; color:black;
              font-family:Calibri; font-size:16px; font-weight:normal;
              text-align:center; margin:5%;}
       h2    {color:darkred;}
       #sbox {background-color:antiquewhite; color:black;
              font-family:Calibri; font-size:20px; font-weight:bold; font-style: italic;
              border: 4px solid darkblue;}
   </style>
   <script>
      function func() {
        var el = document.getElementById("sbox");
        var n  = el.selectedIndex;
        var s  = el.options[n].text.trim().toLowerCase();
        location = s + '.html';
        el.value = "Bitte wählen";
      }
   </script>
</head>

<body>
<form>
   <h2> Eine Auswahlbox im Formular </h2>
   <br>
    <select id="sbox" onchange="func()">
      <option selected disabled hidden>Bitte wählen</option>
      <option>     Demo1     </option >
      <option>     Demo2     </option>
      <option>     Demo3     </option>
    </select>
   <br>
</form>
</body>
</html>
```

(8) demo8.html – Beispiel einer Auswahlliste



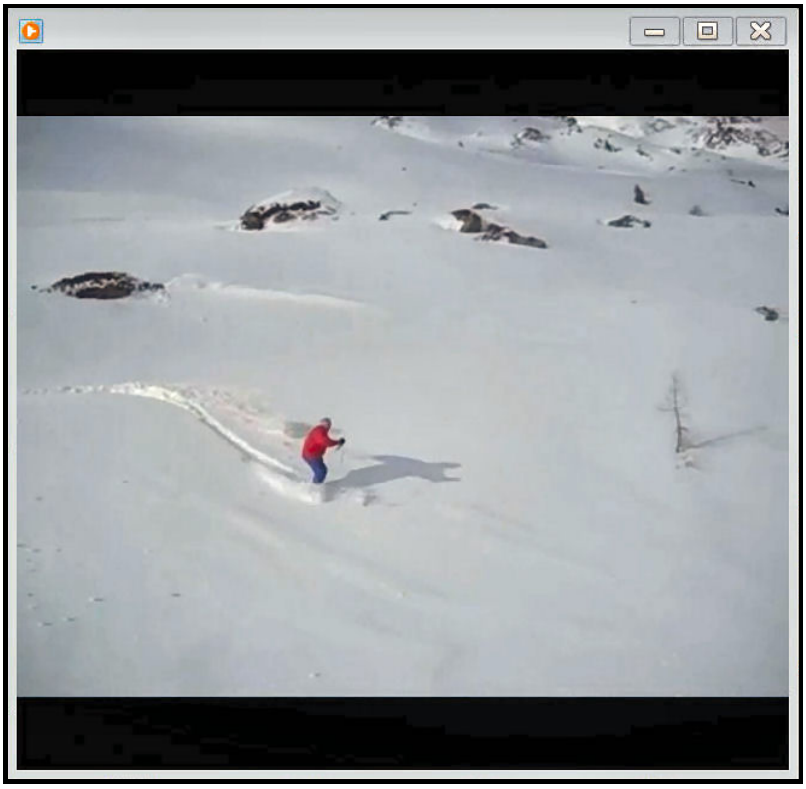
```
<!DOCTYPE html>
<html>
<head>
<title> Auswahllisten  </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" Auswahllisten ">

<style>
body { background-image: url("back1.jpg"); background-color:#CCDDEE; color:black;
       font-family:Calibri,Arial; font-size:20px; font-weight:normal;  }
a:link {color: blue;}
a:visited {color: blue;}
a:hover {color: red;}
span {color: yellow;}
#zentral {
  position:absolute;
  top:45%;
  left:45%;
  width:40em;
  height:20em;
  margin-left:-15em;
  margin-top:-10em;
  border: 10px solid gray;
}
#cell { background-color:antiquewhite; padding:30px; text-align:left; font-weight:bold; }
.menue { background-color:darkblue; color:white; font-style:italic; }
</style>
</head>

<body>
<table id = "zentral">
<tr>
<td id = "cell">

  <p class = "menue">
    Mehrstufige Auswahlliste  
  </p>

  <ol start="1">
    <li><a href="demo1.html">demo1</a></li>
    <li><a href="demo2.html">demo2</a></li>
    <li><a href="demo3.html">demo3</a></li>
    <li><a href="..."> . . . </a></li>
    <li><a href="..."> . . . </a></li>
    <li><a href="..."> . . . </a></li>
    <li><a href="..."> . . . </a></li>
    <li><a href="..."> . . . </a></li>
    <li><a href="..."> . . . </a></li>
    <li><a href="..."> . . . </a></li>
  </ol>

  <p class = "menue">
       Seite 1 von 5
        
     <a href="..."><span> zurück </span></a>
        
     <a href="..."><span> weiter </span></a>
        Seite
     <a href="demo8.html"><span> 1 </span></a>
     <a href="..."><span> 2 </span></a>
     <a href="..."><span> 3 </span></a>
     <a href="..."><span> 4 </span></a>
     <a href="..."><span> 5 </span></a>
  </p>

</td>
</tr>
</table>

</body>
</html>
```

(9) demo9.html – Multimedia mit Text, Bild und Ton




```
<!DOCTYPE html>
<html>
<head>
<title> Multimedia </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" Multimedia ">

<style>
body   { background-image: url(back1.jpg); background-attachment: fixed;
         background-color: #CCDDEE; color: black; margin-left: 5%; }
.bild  { width: 400px; border: 5px solid gray; float: left;}
.block { background-color: #FFFFFF; color: black; width: 360px;
         font-family: Times Roman; font-size:18px; font-weight: normal;
         border: 5px solid gray; padding: 20px; margin: 20px; float: left; }
</style>
</head>

<body>
<br>
<audio width="320" height="240" controls>
    <source src="briefmark.mp3" type="audio/mp3">
    Your browser does not support the audio tag.
</audio>
<br><br>
<img src = "mann.jpg" class="bild">
<br><br>
<div class="block">
    DER BRIEFMARK<br>
    <br>
    Ein männlicher Briefmark erlebte<br>
    was Schönes bevor er klebte.<br>
    Er war von einer Prinzessin beleckt,<br>
    da war die Liebe in ihm erweckt.<br>
    <br>
    Er wollte sie dann wiederküssen,<br>
    da hatte er verreisen müssen.<br>
    So liebt er sie vergebens.<br>
    Das ist die Tragik seines Lebens.<br>
</div>
<br>
</body>
</html>
```

(10) demo10.html – Multimedia mit Video



```
<!DOCTYPE html>
<html>
<head>
<title> Video  </title>
<meta charset="ISO-8859-1">
<meta name="author" content="Paukert Herbert">
<meta name="description" content=" Video ">

<style>
    body { background-color: #667788; margin: auto; }
</style>
</head>

<body>
<br><br>
<video width="640" controls>
    <source src="schi.mp4" type="video/mp4">
    Your browser does not support the video tag.
</video>
<br><br>
</body>
</html>
```

Teil C: Responsives Webdesign (RWD)

Unter **Responsivem Webdesign** (RWD) versteht man die automatische Anpassung der HTML-Objekte an den jeweiligen Geräte-Bildschirm (device-screen) von desktops, laptops, tablets und phones.
Die erste, wichtigste Anweisung dazu ist der HTML-Meta-Tag für den Darstellungsbereich (viewport):

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Diese Anweisung ermöglicht die Breite des Bildschirms in geräteunabhängigen Pixeln zu nutzen. Dadurch können die Inhalte einer Seite neu angeordnet und so an verschiedene Bildschirmgrößen angepasst werden, egal ob an das kleine Display eines Mobiltelefons oder den großen Bildschirm eines Desktopcomputers.

Die zweite Anweisung ist ein CSS-Befehl, welcher die Breite des jeweiligen Geräte-Bildschirms abfragt und dann CSS-Regeln für die Benutzung angibt: **@media (query) { CSS-Rules … }**
Drei Beispiele dazu:

```
@media (max-width: 640px) {
  div { background-color: red; }
}

@media (min-width: 641px) and (max-width: 960px) {
  div { background-color: green; }
}

@media (min-width: 961px) {
  div { background-color: blue; }
}
```

Im ersten Beispiel wird die Hintergrundfarbe eines div-Bereiches auf rot gesetzt, wenn der Browser weniger als oder gleich 640 Pixeln breit ist.
Im zweiten Beispiel wird die Hintergrundfarbe eines div-Bereiches auf grün gesetzt, wenn der Browser zwischen 640 und 961 Pixeln breit ist.
Im dritten Beispiel wird die Hintergrundfarbe eines div-Bereiches auf blau gesetzt, wenn der Browser mehr als oder gleich 961 Pixeln breit ist.

Anmerkung: Auch können Anweisungen wie **@media only screen and (max-width: 640px) { . . . }** verwendet werden.

Das Grundkonzept des responsiven Webdesign ist die flexible Anpassung an die unterschiedlichen Bildschirmgrößen (screens).

Grundsätzlich werden in allen Browsern die verschiedenen Seiteninhalte ohne Verwendung von CSS immer untereinander dargestellt. Bei großen Bildschirmen werden die Inhalte mithilfe von CSS häufig nebeneinander angeordnet. Bei kleinen Bildschirmen müssen die Inhalte mithilfe von CSS hingegen untereinander angeordnet werden, damit nicht oftmalige Verschiebungen und Größenänderungen der Objekte notwendig sind. Daher sollte – wenn möglich – das Layout zuerst für kleine Bildschirme erstellt werden („mobile first“) und dann die Erweiterungen auf große Bildschirme. Der umgekehrte Weg ist meistens mühevoll und kompliziert. Aus diesen Grundsätzen ergeben sich nachfolgende Richtlinien.

- Verwendung relativer Maßeinheiten (anstelle von fixen):
  Angabe der Objektgrößen relativ zur Screenbreite (z.B. width: 100%).
  Angabe der Schriftgrößen relativ zur Standardschrift (z.B. font-size: 1.25em).
  Optimale Lesebarkeit ist mit ca. 80 Zeichen pro Textzeile gegeben,
  wobei {font-family: sans-serif; font-size: 16px} der Schriftstandard ist.

- Möglichst wenige absolute bzw. fixe Positionierungen:
  z.B. display: block; position: relative; float: left.

- Setzen von erforderlichen, inhaltsbezogenen Breakpoints:
  Wenn es der Inhalt einer Seite (page content) erfordert, dann können mit geeigneten „Media Queries“ neue Layout-Richlinien erzeugt werden (so genannte Breakpoints).

Im Folgenden sollen fünf einfache HTML-Programme das responsive Webdesign demonstrieren.

RWD-Demo 1 (respons.html)

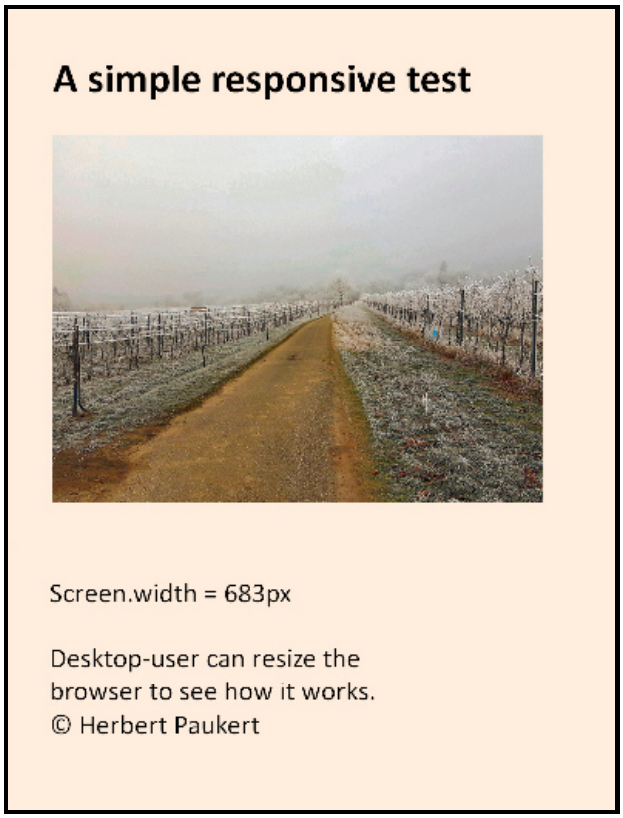
Die Abbildung zeigt die HTML-Seite auf einem großen Bildschirm.

Mit der Hilfe von drei „Media Queries“ werden für phones, tablets und desktops die Hintergrundfarben des Dokumentes, die Imagegrößen eines Bildes und auch die Schriftgröße verändert. Desktop-User können durch „Resizing“ des Screens die Wirkungen direkt beobachten.

```
<!DOCTYPE html>
<html>
<head>
<title>Respons</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Respons">
<meta name="author" content="Paukert herbert">

<style>

* { margin: 0; padding: 10px; }

html { background-color: #DDEEFF; font-family: Calibri,sans-serif; font-size: 16px }
```

```
@media only screen and (max-width: 640px) {
  /* phones */
  html { background-color: #EEDDFF; font-size: 20px; }
  .myImg { width: 70%; }
}

@media only screen and (min-width: 641px) and (max-width: 960px) {
  /* tablets */
  html { background-color: #FFEEDD; font-size: 18px; }
  .myImg { width: 50% }
}

@media only screen and (min-width: 961px) {
  /* desktops */
  html { background-color: #DDEEFF; font-size: 16px; }
  .myImg { width: 30% }
}
</style>
</head>

<body onresize = "showSize()";>
  <br>
  <h2>A simple responsive test</h2>
  <image src="foto.jpg" class="myImg">
  <br>
  <p id="info"> </p>
  <p>
  Desktop-user can resize the<br>
  browser to see how it works.<br>
  &copy Herbert Paukert
  </p>
  <script>
     function showSize() {
       document.getElementById("info").innerHTML = "Screen.width = " + screen.width + "px";
     }
  </script>
</body>
</html>
```

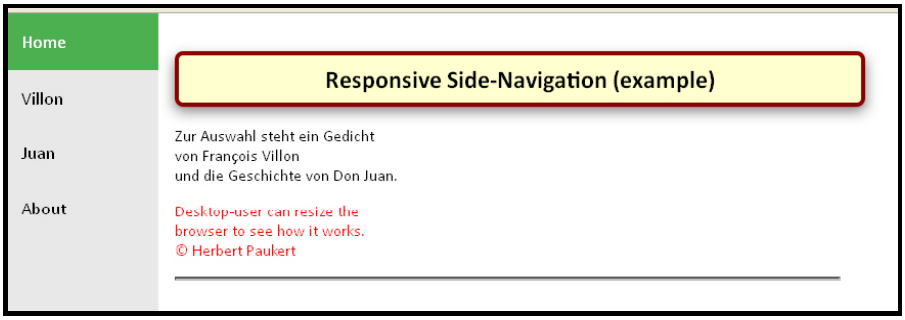
Die Abbildung zeigt die HTML-Seite auf einem kleinen Bildschirm.



RWD-Demo 2 (sidenavi1.html)

Der „**body**“ der HTML-Seite enthält einen dekorierten Titel und darunter normalen Text. Im linken Seitenteil ist ein vertikales Navigationsmenü (**sidebar**) fixiert. Dieses wird durch entsprechende CSS-Attribute erzeugt. Das Menü besteht aus vier Links, wobei der erste und der vierte deaktiviert sind.

Wenn der Bildschirm weniger als 640 Pixel breit ist (**@media screen and (max-width: 640px)**), dann werden alle Links auf die ganze Breite des Bildschirms ausgedehnt. Dabei sind alle Links untereinander angeordnet.



Die vorangehende Abbildung zeigt die Seite für große Bildschirme.
Die nachfolgende Abbildung zeigt die umgewandelte Seite für kleine Bildschirme.


```
<!DOCTYPE html>
<html>

<head>
<title>Side-Navigation 1</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Side-Navigation 1">
<meta name="author" content="Paukert Herbert">

<style>
body   { margin: 0; font-family: Calibri, sans-serif; font-size: 16px; }
```

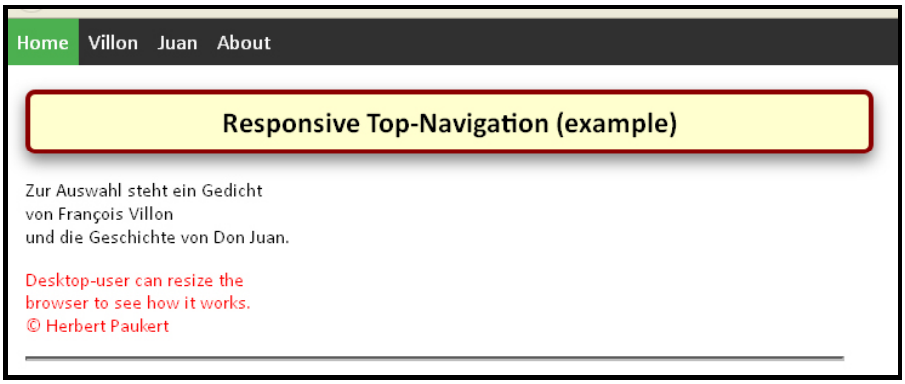
```
#titel {
  width: 55%;
  background-color: #FFFFD0;
  padding: 8px;
  border: 4px solid darkred; border-radius: 8px;
  box-shadow: 0 6px 12px 0 rgba(0,0,0,0.5);
  text-align: center;
}
#satz { color: red; }
hr {
  display: block;
  width: 55%;
  margin-left: 0;
  margin-right:0;
  border-width: 2px;
}
div.content {
  margin-left: 150px;
  padding: 16px;
  height: 100%;
}
.sidebar {
  font-size: 18px;
  margin: 0;
  padding: 0;
  width: 150px;
  background-color: #E8E8E8;
  position: fixed;
  height: 100%;
  overflow: auto;
}
.sidebar a {
  display: block;
  color: black;
  padding: 16px;
  text-decoration: none;
}
.sidebar a.active { background-color: #4CAF50; color: white; }
.sidebar a:hover:not(.active) { background-color: #505050; color: white; }

@media screen and (max-width: 640px) {
  div.content {margin-left: 0;}
  .sidebar {
    width:  100%;
    height: auto;
    position: relative;
  }
  .sidebar a {
    text-align: center;
    float: none;
  }
}
</style>
</head>

<body>
<div class="sidebar">
  <a href="#home" class="active">Home</a>
  <a href="villon.html">Villon</a>
  <a href="donjuan.html">Juan</a>
  <a href="#about">About</a>
</div>
<div class="content">
  <h2 id="titel">Responsive Side-Navigation (example)</h2>
  <p>
  Zur Auswahl steht ein Gedicht<br>
  von François Villon<br>
  und die Geschichte von Don Juan.
  </p>
  <p id="satz">
     Desktop-user can resize the<br>
     browser to see how it works.<br>
     &copy Herbert Paukert
  </p>
  <hr>
</div>
</body>
</html>
```

RWD-Demo 3 (topnavi.html)

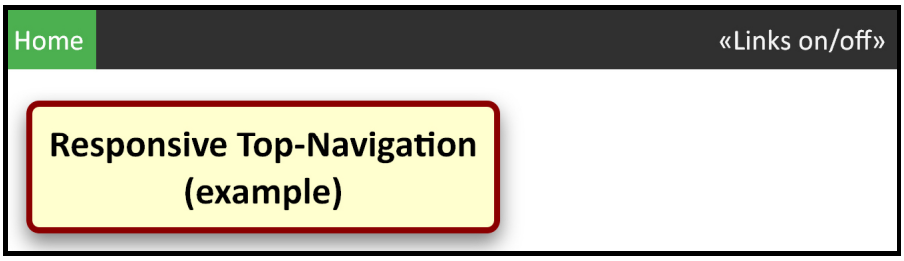
Der „**body**“ der HTML-Seite enthält einen dekorierten Titel und darunter normalen Text. Im Kopfteil der Seite ist ein horizontales Navigationsmenü (**topnav**) fixiert. Dieses wird durch entsprechende CSS-Attribute erzeugt. Das Menü besteht aus vier Links, wobei der erste und der vierte deaktiviert sind.

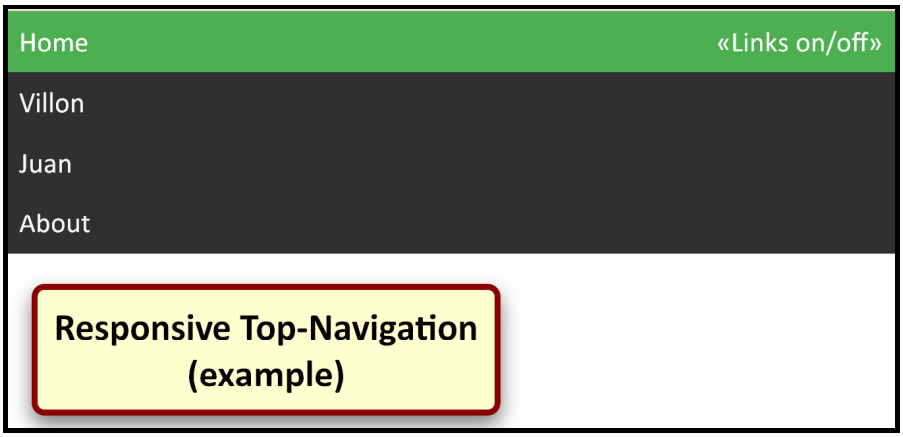
Wenn der Bildschirm weniger als 640 Pixel breit ist (**@media screen and (max-width: 640px)**), dann werden alle vier Links auf die ganze Breite des Bildschirms ausgedehnt und untereinander angeordnet. Dabei werden, ausgenommen den ersten Link, die restlichen Links nicht angezeigt. Stattdessen wird ein Wechselschalter (**Links on/off**) sichtbar. Mit seiner Hilfe können alle Links abwechselnd ein- oder ausgeblendet werden. Diese Klickfunktion wird durch einen JavaScript-Code erzeugt (**myFunction**).

Im vorliegenden Programm wird bei Unterschreiten einer bestimmten Screen-Breite das feste, horizontale Auswahlmenü in ein vertikales PopUp-Menü umgewandelt.



Die vorangehende Abbildung zeigt die Seite für große Bildschirme.
Die nachfolgenden Abbildungen zeigen die umgewandelte Seite für kleine Bildschirme.





```
<!DOCTYPE html>
<html>
<head>
<title>Top-Navigation</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Top-Navigation">
<meta name="author" content="Paukert Herbert">

<style>
body   { margin: 0; font-family: Calibri, sans-serif; font-size: 16px; }
#titel {
  width: 50%;
  background-color: #FFFFD0;
  padding: 8px;
  border: 4px solid darkred; border-radius: 8px;
  box-shadow: 0 6px 12px 0 rgba(0,0,0,0.5);
  text-align: center;
}
#satz { color: red; }
hr {
  display: block;
  width: 50%;
  margin-left: 0;
  margin-right:0;
  border-width: 2px;
}

.topnav { overflow: hidden; background-color: #303030;  }
.topnav a {
  float: left;
  display: block;
  color: #F8F8F8;
  text-align: center;
  padding: 8px;
  text-decoration: none;
  font-size: 18px;
}
.topnav a:hover  { background-color: #D8D8D8; color: black; }
.topnav a.active { background-color: #4CAF50; color: white; }
.topnav .info    { display: none; }

@media screen and (max-width: 640px) {
  .topnav a:not(:first-child) { display: none; }
  .topnav a.info {
    float: right;
    display: block;
  }
}

@media screen and (max-width: 640px) {
  .topnav.responsive { position: relative; }
  .topnav.responsive .info {
    position: absolute;
    right: 0;
    top: 0;
  }
  .topnav.responsive a {
    float: none;
    display: block;
    text-align: left;
  }
}
</style>
</head>

<body>

<div class="topnav" id="myTopnav">
  <a href="#home" class="active">Home</a>
  <a href="villon.html">Villon</a>
  <a href="donjuan.html">Juan</a>
  <a href="#about">About</a>
  <a href="javascript:void(0);" class="info" onclick="myFunction()">
  «Links on/off»
  </a>
</div>
```

```
<div style="padding-left:16px">
  <h2 id="titel">Responsive Top-Navigation (example)</h2>
  <p>
  Zur Auswahl steht ein Gedicht<br>
  von François Villon<br>
  und die Geschichte von Don Juan.
  </p>
  <p id="satz">
  Desktop-user can resize the<br>
  browser to see how it works.<br>
  &copy Herbert Paukert
  </p>
  <hr>
</div>

<script>
function myFunction() {
  var x = document.getElementById("myTopnav");
  if (x.className === "topnav") { x.className += " responsive"; }
  else {  x.className = "topnav"; }
}
</script>

</body>
</html>
```

RWD-Demo 4 (homeres1.html)

Im nachfolgenden Programm wird eine einfache Homepage erzeugt,
deren „**body**“ als Hauptcontainer aus verschiedenen Abschnitten (**divs**) besteht.
Am Seitenanfang befindet sich ein „**header**“ und am Seitenende ein
„**footer**“. Der Abschnitt dazwischen wird in **drei Teile** (**divs**) eingeteilt.
In dem linken Teil befindet sich ein Menü (**menu**), welches aus 4 Links besteht.
Der mittlere und der rechte Teil enthalten verschiedene Texte.
Diese drei Elemente sind von einem Hüllelement (wrapper) umschlossen,
dessen CCS-Attribut { **overflow: auto;** } bewirkt, dass auch über die Elemente
hinaus ragende Inhalte dargestellt werden. Mit { **overflow: hidden;** } werden
sie hingegen abgeschnitten.

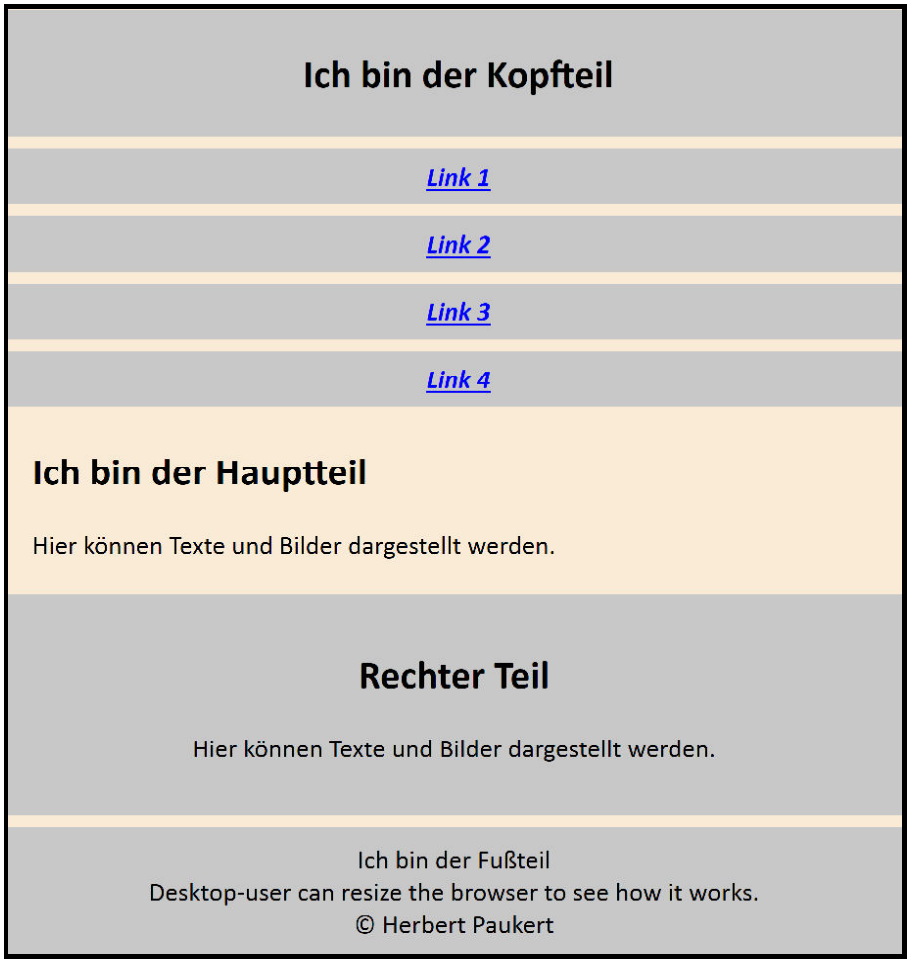
Der Universalselektor * wählt alle Elemente der vorliegenden Seite aus,
und * **{ box-sizing: border-box; }** bewirkt, dass "padding" und "border"
in die Breite (width) und Höhe (height) der Elemente eingerechnet werden.

Die vorangehende Abbildung zeigt die HTML-Seite für große Bildschirme.

Die beschriebene Einteilung der Seite gilt nur für Bildschirme mit einer Mindestbreite von 640 Pixel. Bei kleineren Screens wird jeder der drei Teile in untereinander liegende Blöcke umgewandelt, die alle genau so breit wie der Screen sind:

```
@media only screen and (max-width: 640px) {
   .menu, .main, .right { width: 100%; }
}
```

Die nachfolgende Abbildung zeigt die umgewandelte Seite für kleine Bildschirme.




```
<!DOCTYPE html>
<html>

<head>
<title>Homeres 1</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Responsive Homepage 1">
<meta name="author" content="Paukert herbert">

<style>

* { box-sizing: border-box; }

body    { background-color: antiquewhite; color: black;
          font-family: Calibri,sens-serif; font-size: 18px; }

.header { background-color:#c8c8c8; padding:4px; text-align:center; }

.footer { background-color:#c8c8c8; padding:12px; text-align:center; margin-top: 8px; }
```

```
.wrapper {overflow: auto; }
.menu {
  float:left;
  width:25%;
  text-align:center;
}
.menu a {
  background-color:#c8c8c8;
  padding:8px;
  margin-top:8px;
  display:block;
  width:100%;
  color: blue;
  font-size: 18px;
  font-weight: bold;
  font-style: italic;
}
.main {
  float:left;
  width:50%;
  padding:0 20px;
}

.right {
  background-color:#c8c8c8;
  float:left;
  width:25%;
  padding:16px;
  margin-top:8px;
  text-align:center;
}

@media only screen and (max-width:640px) {
  /* For mobile phones: */
  .menu, .main, .right { width:100%; }
}
</style>
</head>

<body>

<div class="header">
  <h2>Ich bin der Kopfteil</h2>
</div>

<div class="wrapper">

  <div class="menu">
    <a href="#">Link 1</a>
    <a href="#">Link 2</a>
    <a href="#">Link 3</a>
    <a href="#">Link 4</a>
  </div>

  <div class="main">
    <h2>Ich bin der Hauptteil</h2>
    <p>Hier können Texte und Bilder dargestellt werden.</p>
  </div>

  <div class="right">
    <h2>Rechter Teil</h2>
    <p>Hier können Texte und Bilder dargestellt werden.</p>
  </div>

</div>

<div class="footer">
  Ich bin der Fußteil
  <br>
  Desktop-user can resize the browser to see how it works.
  <br>
  &copy Herbert Paukert
  <br>
</div>

</body>
</html>
```

RWD-Demo 5 (homeres2.html)

Zuerst einige Anmerkungen zu den unterschiedlichen CSS-Selektoren, die im Programm verwendet werden.

Grundsätzlich gibt es drei Arten von Selektoren für die Elemente: id-Selektoren (**#id**), Klassen-Selektoren (**.class**), Pseudo-Selektoren (**:**). Der Universalselektor * wählt alle Elemente der vorliegenden Seite aus, und *** { box-sizing: border-box; }** bewirkt, dass "padding" und "border" in die Breite (width) und Höhe (height) der Elemente eingerechnet werden.

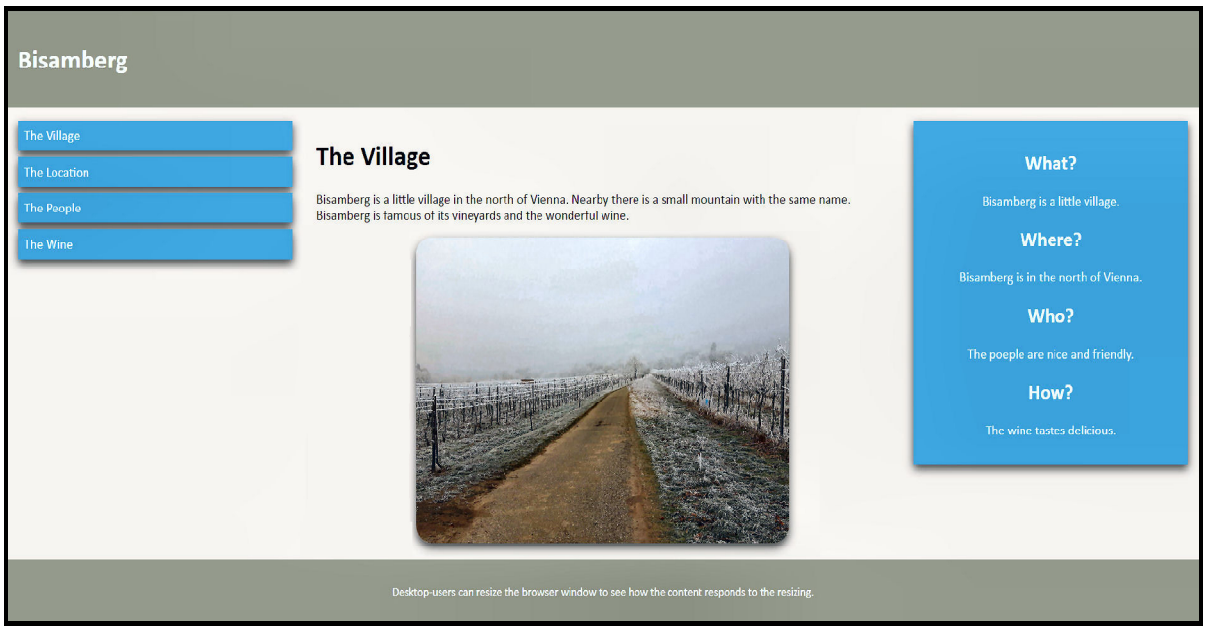
Der Attribut-Selektor **[attr^="wert"]** wählt nur jene Elemente aus, deren Attribute "**attr**" mit dem Ausdruck "**wert**" beginnen.

Der Pseudo-Selektor **:not(.class)** wählt nur jene Elemente aus, welche das nachfolgende Klassenattribut nicht aufweisen.
Die Pseudo-Selektoren **.class::before** oder .**class::after** fügen einen Inhalt (**content**) vor oder nach dem Element mit dem Klassenselektor ein.
Mit **clear:left** wird das Umfließen von Elementen (**float:left**) beendet und das nachfolgende Element wird an den linken Rand gesetzt.

"**display:inline**", "**display:block**" oder "**display:table**", "**display:none**" bestimmen, ob Elemente nebeneinander, untereinander oder überhaupt nicht angezeigt werden. Andere Attribute bestimmen die Stilmerkmale der Elemente (Abstände, Ränder, Größen, Farben, Dekorationen, usw.).

Die Selektoren können auch miteinander kombiniert werden.

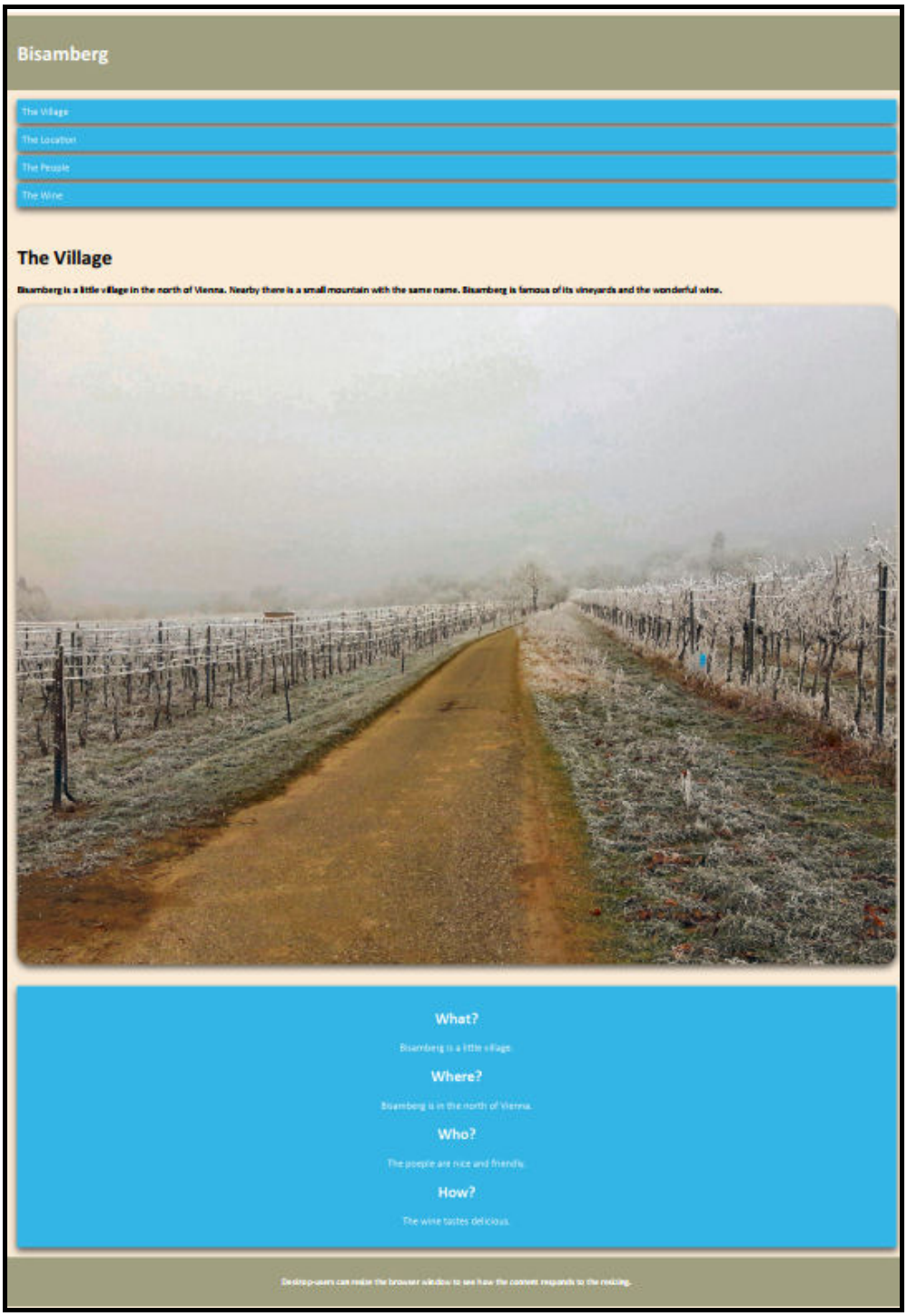
Im nachfolgenden Programm wird eine einfache Homepage erzeugt, deren „**body**“ als Hauptcontainer aus verschiedenen Abschnitten besteht. Am Seitenanfang befindet sich ein „**header**“ und am Seitenende ein „**footer**“. Der Abschnitt dazwischen wird in **drei Spalten** (**cols**) eingeteilt. In der linken Spalte befindet sich eine Menü-Liste (**menu**), die aus 4 Links besteht. Die mittlere Spalte enthält einen Text und darunter ein Bild.
In der rechten Spalte ist ein **aside**-Element eingebettet, wo ein Text steht.



Die vorangehende Abbildung zeigt die Seite für große Bildschirme.

Die beschriebene Spalteinteilung der Seite gilt nur für Bildschirme mit einer Mindestbreite von 640 Pixel (**@media only screen and (min-width: 640px)**). Bei kleineren Screens wird jede der drei Spalten in untereinander liegende Blöcke umgewandelt, die alle genau so breit wie der Screen sind (**[class*="col-"] { width: 100%; }**).

Die nachfolgende Abbildung zeigt die umgewandelte Seite für kleine Bildschirme.



```
<!DOCTYPE html>
<html>

<head>
<title>Homeres 2</title>
<meta charset="ISO-8859-1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Responsive Homepage 2">
<meta name="author" content="Paukert herbert">


<style>

* { box-sizing: border-box; }

html { font-family: Calibri, sans-serif; font-size: 16px; }

body { background-color: antiquewhite; }

.row:after {
  content: "";
  clear: both;
  display: table;
}

#myFoto {
   width: 100%;
   border-radius: 20px;
   box-shadow: 0 5px 10px rgba(0,0,0,0.25), 0 5px 10px rgba(0,0,0,0.50);
 }

[class*="col-"] {
  float: left;
  padding: 15px;
}

.header {
  background-color: #a0a080;
  color: #ffffff;
  padding: 15px;
}

.menu ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}

.menu li {
  padding: 8px;
  margin-bottom: 7px;
  background-color: #33b5e5;
  color: #ffffff;
  box-shadow: 0 5px 10px rgba(0,0,0,0.25), 0 5px 10px rgba(0,0,0,0.50);
}

.menu li a { color: #ffffff; }
.menu li:hover { background-color: #0088aa; }

.aside {
  background-color: #33b5e5;
  color: white;
  padding: 15px;
  text-align: center;
  font-size: 16px;
  box-shadow: 0 5px 10px rgba(0,0,0,0.25), 0 5px 10px rgba(0,0,0,0.50);
}

.footer {
  background-color: #a0a080;
  color: #ffffff;
  text-align: center;
  font-size: 14px;
  padding: 15px;
}

/* For mobile phones: */
[class*="col-"] { width: 100%; }
```

```
@media only screen and (min-width: 640px) {
  /* For tablets: */
  .col-s-1 {width: 8.33%;}
  .col-s-2 {width: 16.66%;}
  .col-s-3 {width: 25%;}
  .col-s-4 {width: 33.33%;}
  .col-s-5 {width: 41.66%;}
  .col-s-6 {width: 50%;}
  .col-s-7 {width: 58.33%;}
  .col-s-8 {width: 66.66%;}
  .col-s-9 {width: 75%;}
  .col-s-10 {width: 83.33%;}
  .col-s-11 {width: 91.66%;}
  .col-s-12 {width: 100%;}
   #myFoto {width: 65%}
}

@media only screen and (min-width: 640px) {
  /* For desktop: */
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
  .col-7 {width: 58.33%;}
  .col-8 {width: 66.66%;}
  .col-9 {width: 75%;}
  .col-10 {width: 83.33%;}
  .col-11 {width: 91.66%;}
  .col-12 {width: 100%;}
   #myFoto {width: 65%}
}
</style>
</head>

<body>
<div class="header">
  <h1>Bisamberg</h1>
</div>

<div class="row">
  <div class="col-3 col-s-3 menu">
    <ul>
      <li>The Village</li>
      <li>The Location</li>
      <li>The People</li>
      <li>The Wine</li>
    </ul>
  </div>

 <div class="col-6 col-s-9">
    <h1>The Village</h1>
    <p> Bisamberg is a little village in the north of Vienna.
        Nearby there is a small mountain with the same name.
        Bisamberg is famous of its vineyards and the wonderful wine. </p>
    <p style="text-align: center;”>
        <img src="foto.jpg" id="myFoto">
    </p>
  </div>

  <div class="col-3 col-s-12">
    <div class="aside">
      <h2>What?</h2>  <p>Bisamberg is a little village.</p>
      <h2>Where?</h2> <p>Bisamberg is in the north of Vienna.</p>
      <h2>Who?</h2>   <p>The poeple are nice and friendly.</p>
      <h2>How?</h2>   <p>The wine tastes delicious.</p>
    </div>
  </div>
</div>
<div class="footer">
  <p> Desktop-users can resize the browser window
      to see how the content responds to the resizing. </p>
</div>

</body>
</html>
```

---------- Schlusswort (in eigener Sache) ----------

Herbert Paukert, der Autor dieses Lehrbuchs – geboren 1945 in Wien und im Berufsleben Lehrer – hat sich 1983/84 als Autodidakt die Maschinensprache und auch die damalige Hochsprache BASIC des ersten guten Kleincomputers „Commodore C64“ angeeignet.

Einige Jahre später waren die Lektüre „Algorithmen und Datenstrukturen“ von *Nikolaus Wirth* und die von ihm erfundene Programmiersprache PASCAL wichtige Schritte. Es folgte dann die Sprache DELPHI mit ihrer integrierten, visuellen Entwicklungsumgebung und dem historischen Höhepunkt im Jahr 2003 mit der Version DELPHI 7 für das Betriebssystem WINDOWS XP. Das Datenbanksystem SQL und die Sprachen C, PHP und HTML waren weitere Etappen.

Die bevorzugte Programmiersprache des Autors war DELPHI 7. Damit programmierte er einen universellen Mathematikparser und ein multimediales Autorensystem zur Erzeugung von Lernsoftware. Er unterrichtete auch zehn Jahre lang Informatik als Dozent am Pädagogischen Institut in Wien.

Erst viele Jahre später, ab 2015/2016 – der Autor war mittlerweile in Pension – erfolgte die intensive Beschäftigung mit der Sprache JAVASCRIPT. Daraus resultierte das hier vorliegende Lehrbuch. Im Februar 2020 hielt der Autor an der Pädagogischen Hochschule in Wien erfolgreich ein Seminar für Lehrer über „Programmieren mit JavaScript“ ab.

Auf Grund der Vielfalt der im Lehrbuch behandelten Themen und Programme ist auch eine gewisse Fehlerwahrscheinlichkeit gegeben. In diesen Fällen ersucht der Autor den interessierten Leser um eine Rückmeldung. Er freut sich aber auch über positive Feedbacks an herbert.paukert@utanet.at.

JavaScript (JS) ist eine Scriptsprache, die 1995 unter *Netscape* von *Brendan Eich* für dynamisches HTML in Webbrowsern entwickelt wurde, um Benutzerinteraktionen auszuwerten, Webinhalte zu verändern, nachzuladen oder zu erzeugen und damit die Möglichkeiten von HTML und CSS zu erweitern.

Zur Datensicherheit wird in JavaScript jede Webanwendung innerhalb des Browsers isoliert in einer sogenannten *Sandbox* ausgeführt. Dies soll bewirken, dass JavaScript nur Zugriff auf die Objekte des Browsers hat und nicht auf das lokale Dateisystem. Einzige Ausnahme ist der Lesezugriff auf lokale Dateien, der über einen Dateiauswahl-Dialog mit dem HTML-Element *<input type = "file">* gestartet wird. Ein Schreibzugriff erfolgt ausschließlich nur in den lokalen Download-Ordner des Benutzers.

Heute ist der Sprachkern von JavaScript durch die *European Computer Manufacturers Association* (**ECMA**) standardisiert und wird im Wesentlichen über **MDN** (*Mozilla Developers Notations*) kontinuierlich weiter entwickelt.

E N D E