Einführung

in

Java-Script

Version 2020.4

Autor: Dieter Lindenberg

Inhaltsverzeichnis

Einleitung ... 3
Aufruf von Java-Script-Programmen ... 5
Kommentare ... 9
Die Dialogbox *confirm* ... 10
Die Dialogbox *prompt* ... 13
Die *while*-Schleife ... 20
Die *do-while*-Schleife ... 21
Die *for*-Schleife ... 22
Eventhandler ... 29
Befehle zur Textverarbeitung ... 33
Zeitverzögerte Methoden ... 45
Laufschriften ... 48
Auslagern von Java-Script-Code ... 52
Die *switch-case*-Anweisung ... 53
Die Klasse *Math* ... 55
Das Objekt *window* ... 67
Das Objekt *document* ... 68
Das Objekt *history* ... 75
Fenster auf der Webseite ... 77
Das Objekt *location* ... 85
Datum und Uhrzeit auf der Homepage ... 87
Bilder ... 96
Formulare auswerten ... 102
Auswahllisten auswerten ... 108
Radiobuttons auswerten ... 109
Checkboxen auswerten ... 111
Frames ... 115
Arrays ... 118
Cascading Style Sheets (CSS) ... 125
Abschalten der rechten Maustaste ... 128
Cookies ... 129
Objekte selbst erzeugen ... 134
HTML-Seite ausdrucken ... 135

Einleitung

Java-Script und *Java* sind völlig getrennt voneinander entstanden, das eine als *Java*, das andere als *LiveScript*.

Die beiden Sprachen haben unterschiedliche Ziele. *Java-Script* wird nur für Webseiten verwendet und auf der Client-Seite (beim Benutzer) ausgeführt. *Java* hingegen wird in der Regel für „normale“ Programme verwendet. Es gibt allerdings auch noch die sog. *Java-Applets*, das sind Programme, die man in Webseiten einbinden kann.

Sun (die "Erfinder" von *Java*) und *Netscape* (die "Erfinder" von *LiveScript*) haben später zusammen kooperiert, damit *Java-Applets* mit *LiveScript* angesteuert werden können. Daraufhin wurde *LiveScript* deswegen in *Java-Script* umbenannt. Bei der Weiterentwicklung von *Java-Script* hat man sich stark an der Programmiersprache Java orientiert.

Java kann man als allgemeine Programmiersprache bezeichnen, *Java-Script* hingegen ist eine sog. *Scriptsprache*, die nur innerhalb eines Browsers ausgeführt werden kann.

Java-Script wird direkt in den HTML-Code geschrieben. Deshalb läuft ein *Java-Script*-Programm auch nur in einem Web-Browser ab.

Leider reagieren verschiedene Browser immer noch unterschiedlich auf *Java-Script*-Befehle. Es kann also durchaus vorkommen, dass ein *Java-Script*-Befehl in einem Browser funktioniert und in einem anderen überhaupt nicht.

Wenn man in einem *Java-Skript*-Editor wie z.B. ***Notepad++*** den Quelltext verändert hat, so muss man ihn erst speichern (und zwar mit dem Namen-anhängsel *.htm oder *.html), bevor man ihn ausführt.
Begründung: ***Notepad++*** führt nur den gespeicherten Quelltext aus.

Java-Script hat große Ähnlichkeit mit der Programmiersprache *Java*. Deshalb sollte man dringend auf Groß- oder Kleinschreibung achten.
Alle *Java-Script*-Schlüsselwörter müssen klein geschrieben werden!
Ausnahmen gibt es nur bei den sog. Event-Handlern (z.B. *onMouseOver*), bei denen es auf Groß-Kleinschreibung nicht ankommt.

Alle *Java-Script*-Befehle sollen mit einem Semikolon beendet werden!

Für die Namen von Variablen und Funktionen gelten folgende Regeln, die in fast allen Programmiersprachen identisch sind. Das erste Zeichen eines Namens muss ein Buchstabe oder ein Dollarzeichen ($) sein. Nachfolgende Zeichen können Buchstaben, Ziffern oder der Unterstrich (_) sein.

Gedankenstriche (= Minuszeichen) sind nicht erlaubt. Diese werden nur für mathematische Subtraktionen benötigt.

Ziffern sind als erstes Zeichen nicht erlaubt. Dadurch kann *Java-Script* einfach zwischen Zahlen und Namen unterscheiden.

Deutsche Umlaute (ä, ö, ü , Ä, Ö, Ü) und der Buchtabe ß sind in Namen von Variablen und Funktionen nicht erlaubt. Dies ist ein bei Programmieranfängern häufig vorkommender Fehler! Programme funktionieren oft überhaupt nicht, nur weil in einem Funktionsnamen Umlaute vorkommen.

Allerdings darf man für bloße Textausgaben durchaus auch Umlaute benutzen. Da die üblichen Browser auf der Welt (und damit auch bei uns in Deutschland) normalerweise keine deutschen Umlaute kennen, könnte man die HTML-üblichen Kodierungen benutzen, also etwa ***ö*** für den Buchstaben ö. Diese Kodierung kann man allerdings nicht innerhalb eines Strings (so heißt der Datentyp für Zeichenketten), also nicht innerhalb von Anführungszeichen einsetzen. Also `"sch&ouml;n"` würde nicht das Wort **schön** darstellen. Innerhalb von Anführungszeichen wird nämlich jedes Zeichen so interpretiert, wie es ist, mit Ausnahme der deutschen Umlaute!

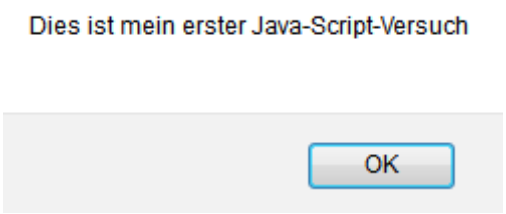
Eine einfache Lösung des Problems ergibt sich, wenn man dem Browser zu Beginn der HTML-Seite mitteilt, dass er sich gefälligst auf den deutschen Zeichensatz einzustellen hat. Dies geschieht so: Man fügt auf der *HTML*-Seite sofort nach dem <head>-Befehl folgende Zeile ein:

```
<meta charset="utf-8">
```

Aufruf von Java-Script-Programmen

Der *Java-Script*-Quelltext kann an einer beliebigen Stelle im HTML-Dokument stehen. Es hat sich jedoch eingebürgert, die *Java-Script*-Befehlszeilen ziemlich am Anfang zu notieren, um eine bessere Übersichtlichkeit zu erhalten. Am besten ist dieser Quelltext zwischen den beiden *Tags* **<head>** und **</head>** aufgehoben. Wir werden jedoch noch sehen, dass einzelne *Java-Skript*-Befehle auch in bestimmten HTML-Elementen verwendet werden.

Im folgenden Beispiel erscheint ein kleines Textausgabefenster auf der Seite.




```
<html>
 <head>
  <script>
   alert("Dies ist mein erster Java-Script-Versuch");
  </script>
 </head>

 <body>
   Hallo Welt!
 </body>
</html>
```

Hinweis: Wenn man Programmtexte aus einer pdf-Datei kopiert (mit der Maus markieren, kopieren und einfügen) und anschließend z.B. in *Notepad++* einfügt, so werden sehr oft die Anführungszeichen nicht richtig übertragen. Das Programm läuft dann nicht wie gewünscht. Abhilfe: Die übertragenen Anführungszeichen müssen in *Notepad++* gelöscht und dann manuell wieder eingefügt werden.

Wenn die *alert*-Dialogbox erscheint, muss der Benutzer erst auf „OK“ klicken, bevor das Programm fortgesetzt wird.

Der obige String *„Dies ist mein ..“* darf im Quelltext nicht durch einen Zeilensprung unterbrochen werden! Begründung: *Java-Script* geht davon aus, dass ein String innerhalb einer (beliebig langen) Zeile abgeschlossen wird. Wenn *Java-Script* kein Ende des Strings erkennt, stürzt das Programm ab.

Es gibt allerdings durchaus die Möglichkeit, mit der *alert*-Anweisung mehrzeilige Texte auszugeben. Dazu benutzt man eine sog. *Escape-Sequenz* **\n**. Beispiel:

```
alert("Dies ist mein erster \nJava-Script-Versuch");
```

Man kann beliebig viele Zeilensprünge ausgeben. Gegebenenfalls erscheint ein Scroll-Balken im *alert*-Ausgabefenster.

Der Browser arbeitet den Quellcode von oben nach unten ab. Also wird in diesem Fall zuerst der *Java-Skript*-Teil ausgeführt, danach der HTML-Teil.

Im folgenden Beispiel bewirkt die *onLoad*-Anweisung, dass **zuerst** der HTML-Teil (einschließlich aller evtl vorhandenen Bilder, Stylesheets und Plugins) geladen und ausgeführt wird und **erst danach** wird eine (von evtl. mehreren) *Java-Script*-Methoden ausgeführt:

```
<html>
 <head>
  <script>
    function willkommen()  {
      alert("Herzlich Willkommen auf meiner Homepage");
    }
  </script>
 </head>

 <body onLoad = "willkommen();">
   Hallo Welt!
 </body>
</html>
```

Natürlich wird auch hier wieder von oben nach unten alles abgearbeitet, aber im Head-Teil wird die *Java-Script*-Funktion nicht aufgerufen. Eine Funktion wird immer durch den **kleingeschriebenen** Begriff ***function*** eingeleitet!

Im letzten Beispiel wurde das *JavaScript*-Programm erst **nach** dem Laden der HTML-Seite aufgerufen. Alternativ könnte der Aufruf dieser Funktion aber auch im Headteil selbst schon stattfinden:

```
<html>
 <head>
  <script>
      willkommen();

      function willkommen()  {
        alert("Sie betreten meine Homepage");
      }
  </script>
 </head>

 <body>
   …..
 </body>
</html>
```

Als weitere Alternative könnte man die Java-Funktion (oder mehrere nacheinander, oder zusätzlich noch weitere Befehle) auch durch einen Link aufrufen:

```
<html>
 <head>
  <script>
    function welcome()  {
      alert("Hallo World !");
    }

    function welcome2()  {
      alert("Hallo World2 !");
    }
  </script>
 </head>

 <body>
  <a href="javascript: welcome(); welcome2();
                              alert('Hallo again');">
    JavaScript aufrufen
  </a>
 </body>
</html>
```

Beachte hier auch das einfache Hochkomma im *alert*-Befehl. Der Grund dafür ist in diesem Fall, dass der gesamte *Java-Script*-Text bereits in normalen (doppelten) Anführungszeichen steht. Man kann Anführungszeichen nicht schachteln.

Kommentare

Längerer *Java-Script*-Code wird üblicherweise am Ende des HEAD-Teils geschrieben.
Java-Skript-Programme werden oft zwischen den **HTML**-Kommentarzeichen <**!**-- und //--> geschrieben.
Begründung: Browser, die kein *Java-Skript* verstehen (weil es zum Beispiel ausgeschaltet worden ist), würden den *Java-Skript*-Programmtext dann einfach als Kommentar ignorieren.
Java-Skript selbst hat die Eigenschaft, dass es unbekannte Befehle (z.B. <**!**--) einfach ignoriert.

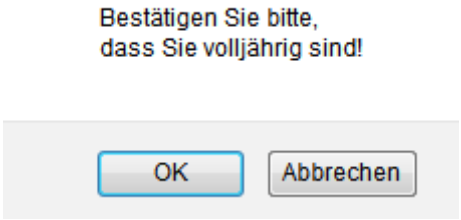
Natürlich gibt es auch in *Java-Skript* Kommentare. Leider werden die anders als in HTML gekennzeichnet:
Einzeilige Kommentare beginnen in ***JavaScript*** mit // . Ein Kommentar-Ende-Zeichen wird hier nicht benötigt, weil am Zeilenende der Kommentar endet.
Mehrzeilige Kommentare müssen hingegen zwischen **/*** und ***/** stehen.

```
<html>
 <head>
  <title> Mein zweiter JavaScript-Versuch </title>
  <!--
  <script>
     Browser, die kein Java-Script verstehen, sehen dies als Kommentar an.
     Java-Skript selbst würde diese Worte hier als unbekannte Befehle
     ebenfalls ignorieren.
     Jetzt würde der Java-Script-Code folgen
  </script>
  //-->
 </head>

 <body>
   .........
 </body>
</html>
```

Die Dialogbox *confirm*

Java-Script verfügt über mehrere vordefinierte Dialogboxen, die grundsätzlich immer in der Mitte des Bildschirms erscheinen.

In dem nächsten Programm wird nur der Befehl *alert* durch *confirm* ersetzt. Dieser neue Befehl gestattet es dem Benutzer, zwischen *'OK'* und *'Abbrechen'* zu wählen. Da wir mittlerweile mehrere Möglichkeiten kennen, ein *Java-Skript*-Programm aufzurufen, wird dieser Programmaufruf ab jetzt nicht mehr mit angegeben.




```
<script>
  function bestaetigung()  {
   confirm("Bestätigen Sie bitte, \ndass Sie volljährig sind!");
  }
</script>
```

Sinnvoll ist der *confirm*-Befehl natürlich nur dann, wenn das Programm auf *'OK'* und *'Abbrechen'* jeweils unterschiedlich reagiert. Es lässt sich leicht unterscheiden, ob der Benutzer auf "OK" oder auf "Abbrechen" gedrückt hat. Der *confirm*-Befehl liefert nämlich ein Ergebnis zurück, genauer: einen sog. Wahrheitswert (*boolean*).
Wenn der Anwender auf "OK" klickt, so hat dieses Ergebnis den Wert "***true***". Wenn er "Abbrechen" wählt (oder im Internet-Explorer das Dialogfenster mit dem Icon X oben rechts schließt), ist das Ergebnis "***false***".

Im nachfolgenden Beispiel wird dieses Ergebnis in einer Variablen namens *eingabe* gespeichert. Beachte dabei auch Folgendes beim Schreiben des Programmes:

- Die *Java-Script*-Worte ***var, true*** und ***if*** müssen klein geschrieben sein!
- Es kommt auf Groß- und Kleinschreibung an: *Eingabe* ≠ *eingabe*
- Für die Variable *eingabe* braucht kein Typ (*String*, *int*, *char* usw.) angegeben zu werden. Begründung: *Java-Script* erkennt bei der ersten Zuweisung eines Wertes an die Variable den Variablentyp. Eventuell notwendige Typumformungen führt *Java-Script* selbständig durch.
- Die Variable *eingabe* ist hier eine **lokale** Variable. **Globale** Variablen sollten direkt hinter dem *Script*-Tag stehen.
- Wird das *Java-Script*-Wort ***var*** vergessen, so wird die entsprechende Variable automatisch als global angesehen.

Um den Gültigkeitsbereich einer Variablen lokal festzulegen, muss innerhalb von Funktionen **`var`** benutzt werden. Fehlt dieses Wort **`var`** bei der Variablendeklaration innerhalb von Funktionen, so ist die entsprechende Variable global gültig. Variablendeklarationen außerhalb von Funktionen sind immer global, unabhängig davon, ob sie mit **`var`** deklariert werden oder nicht.
Allgemein gilt also: Ohne **`var`** ist eine Variablendeklaration immer global.

Das Verwenden von **`var`** gilt auch, wenn man mehrere Variable deklariert.
Beispiel: Die Befehlszeile **`var name, i, x;`** legt innerhalb einer Funktion alle drei Variablen als lokal fest.

Hinweis: Die Schreibweise **`Var`** wird von Java-Script nicht als **`var`** erkannt!

- Das Gleichheitszeichen hat hier unterschiedliche Bedeutung: Wenn etwas **definiert** (zugewiesen) werden soll, schreibt man nur `"="`.
  Wenn etwas **verglichen** werden soll, schreibt man zwei Gleichheitszeichen hintereinander.
- Die Bedingung in der *if-Abfrage* muss in Klammern stehen!
- Hinter der *Bedingung* in der *if-Abfrage* darf kein Semikolon stehen, weil ein Semikolon nur hinter einem Befehl steht. Wenn vor einem Semikolon kein Befehl steht, dann wird auch nichts gemacht.
- Der gesamte *else-Teil* kann auch fehlen.
- Wenn in einem Block nur ein einziger Befehl steht, so dürfen die geschweiften Klammern auch fehlen.

```
<script>
 function duSie()
   {
     var eingabe;
     eingabe = confirm("Sind Sie damit einverstanden,
                                    \ndass wir Sie duzen?");
     if (eingabe == true)
       {
        alert("Hallo du!");
       }
     else
        {
          alert("Hallo Sie!");
        }
   }
</script>
```

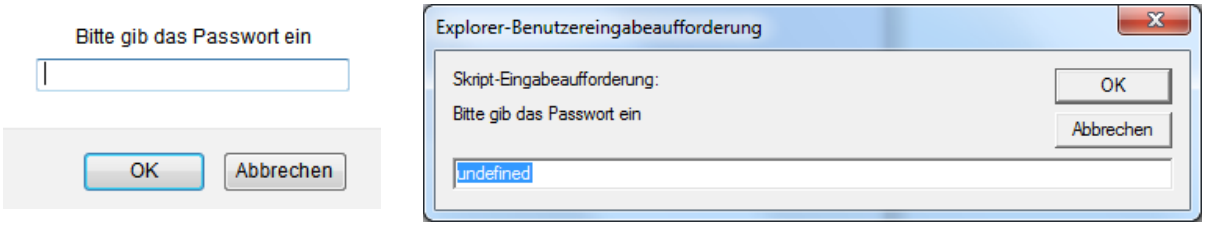
Im obigen Programm wird die lokale Variable *eingabe* zuerst **deklariert** und danach in der nächsten Zeile erst **definiert**. In vielen Programmiersprachen muss bei der Deklaration noch der sog. Datentyp (Text, Zahl, Wahrheitswert) angegeben werden. Man kann allerdings auch Deklaration und Definition in einem einzigen Schritt zusammen ausführen (siehe folgende Beispiele!).

Obiges Programm ließe sich verkürzt auch so schreiben:

```
<script>
 function duSie()
   {
     var eingabe = confirm("Sind Sie damit einverstanden,
                                    \ndass wir Sie duzen?");
     if (eingabe == true) alert("Hallo du!");
     else alert("Hallo Sie!");
   }
</script>
```

Die Dialogbox *prompt*

Im folgenden Programm muss der Internet-Surfer ein Passwort eingeben. Je nach der Richtigkeit der Eingabe reagiert das Programm unterschiedlich.




```
<script>
  function passwort()  {
     var eingabe = prompt("Bitte gib das Passwort ein");
     if (eingabe=="Informatik" || eingabe=="Lindenberg")
        window.open("Seite2.htm");
     else alert("falsches Passwort!");
  }
</script>
```

Normalerweise gibt man bei der *prompt*-Methode auch einen Eingabevorschlag mit aus:

```
<script>
   function fach()  {
     var eingabe = prompt("Bitte gib dein
                          Lieblingsfach ein", "Musik");
     alert("Dein Lieblingsfach ist also " + eingabe);
  }
</script>
```

Die Methode ***prompt()*** zeigt ein Dialogfenster mit einem Eingabefeld, einem OK-Button und einem Abbrechen-Button an. Der Anwender kann in dem Eingabefeld etwas eingeben. Die Methode ***prompt()*** gibt diesen eingegebenen Wert zurück. So lassen sich Anwendereingaben im Script weiterverarbeiten. Man muss einen oder zwei Parameter dabei angeben:

1. Aufforderungstext = Text, der beschreibt, was der Anwender eingeben soll.
2. (optional) Feldvorbelegung = Text, mit dem das Eingabefeld vorbelegt wird. Wenn man ein leeres Eingabefeld will, übergibt man dafür die sog. **leere Zeichenkette** **""**.

Das Ergebnis der prompt-Methode wird zurückgeliefert und kann in einer Variablen gespeichert werden. Falls der Benutzer auf "OK" klickt, wird der von ihm eingegebene Text (evtl. der sog. **Leerstring**) zurückgegeben. Falls er "Abbrechen" wählt, wird als Ergebnis ***null*** zurückgegeben.

Im ersten obigen Beispiel werden bei der *if-Abfrage* zwei Bedingungen durch das logische ODER verknüpft. Die logischen Operatoren werden in Java-Script folgendermaßen realisiert:

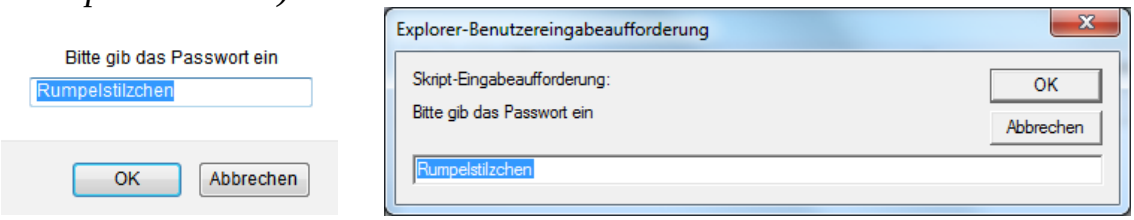
ODER	**||**
UND	**&&**
NOT	**!**

Für diese drei Operatoren gibt es auch eine Prioritäten-Regelung (ähnlich wie in der Mathematik: Potenzrechnung vor Punktrechnung vor Strichrechnung):
NOT vor **UND** vor **ODER**
Gegebenenfalls muss man deshalb auch, wie in der Mathematik, Klammern setzen.

Als Vergleichsoperatoren kennt man die Zeichen **<, >, <=, >=, = =, !=**

Der Befehl ***window.open*** öffnet ein neues Fenster.

Im Eingabefeld der obigen Dialogbox erscheint (im Microsoft-Internet-Explorer) als voreingestellter Wert das Wort "*undefined*". Man kann jedoch auch einen Eingabewert vorschlagen: *prompt("Bitte gib das Passwort ein", "Rumpelstilzchen")*.



Möchte man auch im Microsoft-Internet-Explorer eine leere Eingabezeile, so schreibt man *prompt("Bitte gib das Passwort ein","")*.

Falls die *prompt*-Dialogbox mit dem Button *Abbrechen* beendet wird (oder im Internet-Explorer das Dialogfenster mit dem Icon X oben rechts geschlossen wird), erhält die Zuweisung den Wert ***null***. Man könnte also im obigen Beispielsprogramm auch folgende Abfrage stellen:

```
<script>
  function kontrolle()  {
    var eingabe = prompt("Passwort eingeben!", "");
    if (eingabe == null)
       alert("Sie müssen ein Passwort eingeben!");
    else if (eingabe=="Goethe-Gymnasium")
             window.open("Seite2.htm");
         else alert("falsches Passwort!");
  }
</script>
```

Bemerkung: Der Befehl *prompt()* lässt sich auch ohne Eingabeaufforderung benutzen. Beispiel: `eingabe = prompt();`

Die im obigen Programm auftretende Textausgabe *„Sie müssen ein Passwort eingeben“* macht eigentlich nur Sinn, wenn man dem Benutzer auch anschließend noch einmal die Möglichkeit gibt, ein Passwort einzugeben. Für eine derartige Programmierung werden üblicherweise sog. Schleifen eingesetzt.

Aufgabe 16-1
Dem Besucher wird eine einfache Multiplikationsaufgabe gestellt. Er soll das Ergebnis eingeben. Der Rechner gibt anschließend „richtig!“ oder „falsch!“ aus.

Aufgabe 16-2
Der Besucher wird aufgefordert, eine natürliche Zahl einzugeben. Falls diese eingegebene Zahl zwischen 20 und 30 (einschließlich der beiden Grenzen) oder zwischen 80 und 90 (einschließlich der beiden Grenzen) liegt, wird „OK“ ausgegeben, ansonsten wird „falsche Zahl!“ ausgegeben.

Aufgabe 16-3
Der Besucher wird zweimal nacheinander aufgefordert, ein Passwort einzugeben. Beim ersten Passwort soll er den Namen der besten Schule eingeben (es wird natürlich *„Goethe-Gymnasium“* erwartet). Beim zweiten Passwort soll er den Namen eines äußerst interessanten Unterrichtsfaches eingeben (akzeptiert wird jedes der drei Worte *Informatik*, *Mathematik* und *Physik*).
Sind beide Passwörter richtig, so wird die Note „sehr gut“ ausgegeben, bei nur einem richtigen Passwort wird „gut“ ausgegeben. Bei zwei falschen Passwörtern wird „ungenügend“ ausgegeben.

Aufgabe 16-4
Der Benutzer erhält (höchstens) drei Mal die Möglichkeit, das richtige Passwort einzugeben (welches *Goethe* lautet). Bei richtiger Passworteingabe wird er sofort zur eigentlichen Seite (mit dem Dateinamen *eigentlicheSeite.html*) weitergeleitet, welche für diese Aufgabe nur einen Willkommensgruß beinhaltet. In der Realität könnte dies natürlich die eigentliche Startseite der Homepage sein.
Bei dreimaliger falscher Passworteingabe erfolgt nur die Meldung „Zutritt verweigert!“

Lösungen

Aufgabe 16-1

```
<html>
 <head>
  <script>
    function matheAufgabe()  {
      var ergebnis = prompt("Wie viel ist 7*8 ?","");
      if (ergebnis == 56)  alert("richtig !");
      else alert("falsch!");
    }
  </script>
 </head>

 <body onLoad = "matheAufgabe()" >
   …..
 </body>
</html>
```

Aufgabe 16-2

```
<script>
  function zahlEingabe()  {
    var n = prompt("Gib eine natürliche Zahl
                                          ein!","");
    if (20<=n && n<=30  ||  80<=n && n<=90  )
      alert("OK");
    else alert("falsche Zahl!");
  }
 </script>
```

Aufgabe 16-3

```
<script>
  function passwortabfrage()  {
    var eingabeSchule = prompt("Bitte den Namen der
                        besten Schule eingeben!", "");
    var eingabeFach = prompt("Bitte den Namen des
                        besten Faches eingeben!", "");

    if (eingabeSchule == "Goethe-Gymnasium" &&
        (eingabeFach == "Informatik" ||
         eingabeFach == "Mathematik" ||
         eingabeFach == "Physik"))
      alert("sehr gut");
    else if (eingabeSchule == "Goethe-Gymnasium" ||
              eingabeFach == "Informatik" ||
              eingabeFach == "Mathematik" ||
              eingabeFach == "Physik")
          alert("gut");
         else alert("ungenügend");
  }   // Ende der Funktion

  </script>
```

Aufgabe 16-4

```
<html>
 <head>
  <meta charset="utf-8">
  <script>
    function passwortabfrage()  {
      var eingabe = prompt("Bitte Passwort eingeben!");
      if (eingabe == "Goethe")
         window.open("eigentlicheSeite.html");
      else {
        eingabe = prompt("Falsches Passwort! Bitte
                                  nochmal eingeben!");
        if (eingabe == "Goethe")
          window.open("eigentlicheSeite.html");
        else {
          eingabe = prompt("Falsches Passwort! Letzte
             Möglichkeit!\nBitte nochmal eingeben!");
          if (eingabe == "Goethe")
            window.open("eigentlicheSeite.html");
          else alert("Zutritt verweigert!");
        }
      }
    }
  </script>
 </head>


 <body onLoad = "passwortabfrage()" >
   ……..
 </body>
</html>
```

Die *while*-Schleife

Die *while*-Schleife wird üblicherweise benutzt, wenn man nicht genau weiß, wie oft etwas gemacht werden soll. Solange die anfängliche Bedingung der *while*-Schleife erfüllt ist, wird die Schleife durchlaufen. Es kann auch vorkommen, dass die Schleife überhaupt nicht durchlaufen wird.
In der folgenden Programmversion wird ein Passwort solange abgefragt, bis es richtig eingegeben wird.

```
<script>
  function kontrolle()  {
    var eingabe;
    while (eingabe != "Rumpelstilzchen")  {
      eingabe = prompt("Bitte gib Passwort ein!","");
      if (eingabe != "Rumpelstilzchen")
        alert("falsches Passwort!");
    }
  }
</script>
```

Jede **Schleife** lässt sich mit dem Befehl *break* abbrechen, wie das folgende Beispiel zeigt. Man kann übrigens einer Variablen auch schon bei der Deklaration einen Anfangswert zuweisen.

```
<script>
  function kontrolle ()  {
    var eingabe = "Willi", i=0;
    while (eingabe != "Rumpelstilzchen")  {
     eingabe = prompt("Bitte gib Passwort ein","");
     if (eingabe != "Rumpelstilzchen")  {
       alert("falsches Passwort!");
       i = i+1;
       if (i == 3)  {
         alert("Du lernst es nie!");
         break;
       }
     } // end of if eingabe….
    } // end of while
    alert("die Passworteingabe wurde beendet");
  } // end of function
</script>
```

Die *do-while*-Schleife

In der folgenden ***do-while* - Schleife** wird der Code der Schleife zuerst einmal ausgeführt und erst danach die Bedingung geprüft. Ist die Bedingung erfüllt, wird die Schleife erneut ausgeführt. Diese Schleife wird also mindestens einmal ausgeführt. Üblicherweise weiß man auch bei der *do-while*-Schleife nicht genau, wie oft die Schleife durchlaufen wird.

```
<script>
   var a = 10;
   do  {
     a++;    // das ist eine Abkürzung für  a=a+1;
     alert(a);
   }
   while (a < 14);
   ......
   ......
</script>
```

Beachte, dass hinter der *while-Bedingung* diesmal ein Semikolon stehen muss, weil es das Ende der *do-while-Schleife* angibt.

Die *for*-Schleife

Die for-Schleife wird meistens dann eingesetzt, wenn man genau weiß, wie oft etwas gemacht werden soll.

```
<script>
  function summe()  {
     var s=0;
     for (var n=1; n<=100; n++)  s = s + n;
     alert(s);
  }
```

Bemerkung: die lokale Laufvariable **n** wird üblicherweise innerhalb der *for*-Schleife deklariert.

```
  function summe2()  {
     var n, s = 0; //unüblicher Ort für die Laufvariable n der Schleife
     for (n=2; n<=200; n=n+2)  s = s+n;
     alert(s);
  }
```

Wenn der Operator + zwischen zwei Texten steht, so werden die beiden Texte einfach hintereinandergehängt.
Beispiel: **"Hans" + "wurst"** ergibt **"Hanswurst"**

Wenn der Operator + zwischen einem Text und einer Zahl steht, so wird zuerst die Zahl in einen Text umgewandelt und danach werden die beiden Texte hintereinandergehängt.
Beispiel: **"BVB 0" + 9** ergibt **"BVB 09"**

```
  function kubikAusgabe()  {
    var ausgabe="";
    for (var i=10;i>0; i--)   // i-- ist eine Abkürzung für i=i-1
       ausgabe = ausgabe + i*i*i + "\n";
    alert(ausgabe);
  }
```

In *for*-Schleifen sollten möglichst immer lokale Variablen benutzt werden. Betrachte die beiden folgenden Beispiele, deren Quellcodes sich nur an einer einzigen Stelle unterscheiden! Es werden jedoch unterschiedliche Ergebnisse geliefert.

```
<script>
   var i = -1;     // Dies ist eine globale Variable
   alert(i);
   summe();

   function summe()  {
      var s = 0;
      for (var i=1; i<=10; i++)
        s = s + i;  // i ist hier eine (andere) lokale Variable
      alert("Summe: " + s);
   }
   alert(i);

</script>
```

Die drei *alert*-Anweisungen im obigen Beipiel liefern nacheinander die drei Ausgaben -1, Summe: 55, -1.

Erklärung: Wenn in einer Funktion eine lokale Variable denselben Namen hat wie ein globale Variable, so wird innerhalb der Funktion nur mit der lokalen Variablen gerechnet.
Nach Beendigung der Funktion existieren die lokalen Variablen nicht mehr.

```
<script>
   var i = -1;
   alert(i);
   summe();

   function summe()  {
      var s = 0;
      for (i=1; i<=10; i++)
        s = s + i;   // i ist hier die globale Variable
      alert("Summe: " + s);
   }
   alert(i);
</script>
```

Die drei *alert*-Anweisungen im obigen Beipiel liefern nacheinander die drei Ausgaben -1, Summe: 55, **11**.

Aufgaben

1. Gib die ersten 10 Quadratzahlen aus! Gib drei unterschiedliche Lösungen an mit jeweils einem anderen Schleifentyp!

2. Gib die Summe der ersten 10 Quadratzahlen aus! Gib drei unterschiedliche Lösungen an mit jeweils einem anderen Schleifentyp!

3. Es wird ein Passwort abgefragt. Der Besucher erhält 5 Chancen, das richtige Passwort einzugeben.

4. Gib deinen Namen n mal nacheinander aus! Die Zahl n wird mit der Methode *prompt* eingelesen.

5. a) Gib alle ganzen Zahlen zwischen 100 und 110 aus!
b) Gib alle ganzen Zahlen zwischen 100 und 110 rückwärts aus!

6. Gib die Fakultät der Zahl n, welche mit der *prompt*-Anweisung eingelesen wird, aus!

7. Berechne die n-te Potenz der Zahl 2 und gib sie aus! Die Zahl n wird wieder mit *prompt* eingelesen.

8. Wie viele natürliche Zahlen zwischen 1 und 1 000 gibt es, die zugleich Quadratzahl und Kubikzahl sind? Beispiel: $64 = 4^3 = 8^2$
Gib all diese Zahlen nacheinander aus!

9. Die Fibonacci-Folge lautet: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ... Der Computer soll die ersten 15 Zahlenglieder berechnen und ausgeben.

10. Der Computer berechnet die Summe der ersten 100 natürlichen Zahlen und gibt sie aus.

11. Der Computer berechnet die Summe der ersten 100 natürlichen Quadratzahlen und gibt sie aus.

12. Der Computer berechnet die Summe der folgenden 100 Brüche und gibt sie aus: $\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{100}$

13. Es wird ein Passwort abgefragt. Akzeptiert werden „Goethe“ und „goethe“.

Lösungen

Aufgabe1

```
<script>
  function aufgabe1While()   {
    var ausgabe = "";
    var i=1;
    while (i<=10)  {
      ausgabe = ausgabe + i*i +"\n";
      i++;
    }
    alert(ausgabe);
 }


 function aufgabe1Do()   {
   var ausgabe = "";
   var i = 1;
   do  {
     ausgabe = ausgabe + i*i +"\n";
     i++;
   }
   while (i<=10);
   alert(ausgabe);
 }


 function aufgabe1For()   {
   var ausgabe = "";
   for (var i=1; i<=10; i++)  ausgabe = ausgabe + i*i +"\n";
   alert(ausgabe);
 }
</script>
```

Aufgabe2

```
<script>
  function aufgabe2While()   {
    var summe = 0;
    var i=1;
    while (i<=10)  {
      summe = summe + i*i;
      i++;
    }
    alert(summe);
  }
```

```
  function aufgabe2Do()    {
    var summe = 0;
    var i=1;
    do  {
      summe = summe + i*i;
      i++;
    }
    while (i<=10);
    alert(summe);
  }

  function aufgabe2For()    {
    var summe = 0;
    for (var i=1; i<=10; i++)  summe = summe + i*i;
    alert(summe);
  }
</script>
```

Aufgabe4

```
 function aufgabe4()  {
   var ausgabe = "";
   var name = "Willi Wacker";
   var anzahl = prompt("Bitte gib die Anzahl  n  ein!", "0");
   // besser und sicherer wäre die Anweisung
   // var anzahl = eval(prompt("Bitte gib die Anzahl  n  ein!", "0"));
   for (var i=1; i<=anzahl; i++)
      ausgabe = ausgabe + name + "\n";
   alert(ausgabe);
 }
```

Aufgabe5a

```
 function aufgabe5a()  {
   var ausgabe = "";
   for (var i=100; i<= 110; i++)
     ausgabe = ausgabe + i + "\n";
   alert(ausgabe);
 }
```

Aufgabe5b

```
 function aufgabe5b()  {
   var ausgabe = "";
   for (var i=110; i>= 100; i--)
     ausgabe = ausgabe + i + "\n";
   alert(ausgabe);
 }
```

Aufgabe6

```
function aufgabe6()  {
  var fak = 1;
  var n = prompt("Bitte gib die Zahl n ein!", "0");
   // besser und sicherer wäre die Anweisung
   // var n = eval(prompt("Bitte gib die Zahl  n  ein!", "0"));
   for (var i=1; i<= n; i++)  fak = fak * i;
   alert(fak);
}
```

Aufgabe7

```
function aufgabe7()  {
  var pot = 1;
  var n = prompt("Bitte gib den Exponenten n ein!", "0");
   // besser und sicherer wäre die Anweisung
   // var n = eval(prompt("Bitte gib den Exponenten  n  ein!", "0"));
  for (var i=1; i<= n; i++)  pot = pot * 2;
  alert(pot);
}
```

Aufgabe8 *(for-Schleife)*

```
function aufgabe8()   {
  var ausgabe = "";
  for (var a=1; a<32; a++)
    for (var b=1; b<=a; b++)
      if (a*a == b*b*b)
        ausgabe =
           ausgabe + a*a+" = " +a+ "^2 = " +b+"^3\n";
  alert(ausgabe);
}
```

Aufgabe9

```
<script>
  function aufgabe9()  {
    var f1 = 0;
    var f2 = 1;
    var f3;
    var ausgabe = "0\n1\n";

    for (var i=3; i<= 15; i++)    {
      f3=f1+f2;
      ausgabe = ausgabe + f3 + "\n";
      f1=f2;
      f2=f3;
    }
    alert(ausgabe);
  }
</script>
```

Aufgabe 10

```
<script>
  var summe = 0;
  for (var i=1; i<=100; i++) summe = summe + i;
  alert(summe);
</script>
```

Aufgabe 11

```
<script>
  var summe = 0;
  for (var i=1; i<=100; i++) summe = summe + i*i;
  alert(summe);
</script>
```

Aufgabe 12

```
<script>
   function Brueche()  {
     var summe = 0;
     for (var i=1; i<=100; i++)  summe = summe + 1/i;
     alert(summe);
   }
  </script>
```

Eventhandler

Eine Webseite kann auf verschiedene Ereignisse reagieren. Diese Ereignisse (engl.: event) sind zum Beispiel Mausklick, Mausdruck, Mausbewegung, Tastendruck. Die sog. *Eventhandler* stellen eine Verbindung zwischen HTML und *Java-Script* her.
Aufgrund dessen werden sie als Attribute in den HTML-*Tags* aufgenommen. Üblicherweise rufen sie dann einzelne *Java-Script*-Funktionen auf. Nicht jeder *Eventhandler* funktioniert in jedem *HTML-Tag*. Das hängt leider auch noch vom Browser und sogar von dessen Versionsnummer ab. Beispiel: der Eventhandler *onBeforeUnload* funktioniert im Jahr 2017 leider nicht in *Firefox*, aber im *InternetExplorer*.
Der Eventhandler kann auch nacheinander mehrere Funktionen aufrufen.

```
<img src = "Bild1.htm"  onClick = "willkommen1();
                                   willkommen2();">
```

Attribut	Bedeutung
onBlur	wenn ein Element den Fokus verliert
onChange	wenn eine Menüauswahl geändert wird
onClick	beim Anklicken des Elements
onDblClick	beim doppelten Anklicken des Elements
onError	bei Fehlermeldung. Bsp.: ein Bild konnte nicht geladen werden
onFocus	wenn ein Feld den Fokus erhält
onMouseDown	bei Drücken der Maustaste über dem Element
onMouseUp	bei Loslassen der Maustaste über dem Element
onMouseOver	bei Überfahren des Elements mit der Maus
onMouseMove	bei Ziehen des Elements mit der Maus
onMouseOut	bei Verlassen des Elements mit der Maus
onKeyPress	bei Drücken und wieder Loslassen einer Taste, (im Body-Tag)
onKeyDown	bei Drücken einer Taste, (im Body-Tag)
onKeyUp	bei Loslassen einer Taste, (im Body-Tag)
onLoad	wird ausgeführt, wenn der HTML-Code vollständig geladen ist
onBeforeUnload	wird ausgeführt, wenn die Seite wieder verlassen wird
onResize	Bei Änderung der Größe des Fensters

Aufgaben

1. Erstelle eine HTML-Seite mit 5 **kleinen** Bildern!

- Ein Click auf das erste Bild bewirkt die Meldung: *„Ich bin das erste Bild“*.
- Ein Doppelklick auf das zweite Bild bewirkt eine ähnliche Meldung.
- Wird auf dem dritten Bild die Maus heruntergedrückt, so erscheint die Meldung: *„ich bin das 3. Bild“*.
- Lässt man über dem vierten Bild die vorher gedrückte Maustaste wieder los, so erscheint die Meldung: *„ich bin das vierte Bild“*.
- Wird die Maus über das fünfte Bild bewegt, so erscheint eine entsprechende Meldung.

2. Mach dir den Unterschied klar zwischen *onClick* und *onMouseDown* !

3. Erzeuge zwei HTML-Seiten. Die zweite Seite soll mit einem Link von der ersten Seite aus erreichbar sein. Wird die zweite Seite wieder verlassen, so erscheint die Meldung *„Auf Wiedersehen“*

4. Wenn man eine Taste drückt, erscheint die Meldung *„Lass die Finger von der Tastatur!“*

Lösungen

Aufgabe1

```
<html>
 <head>
  <script>
     function A1()  {
       alert("ich bin das erste Bild");
     }

     function A2()  {
       alert("ich bin das zweite Bild");
     }

     function A3()  {
       alert("ich bin das dritte Bild");
     }

     function A4()  {
       alert("ich bin das vierte Bild");
     }

     function A5()  {
       alert("ich bin das fuenfte Bild");
     }

  </script>
 </head>

 <body>
   <img src = "bildName1.gif" onClick = "A1()">
   <img src = "bildName2.gif" onDblClick = "A2()">
   <img src = "bildName3.gif" onMouseDown = "A3()">
   <img src = "bildName4.gif" onMouseUp = "A4()">
   <img src = "bildName5.gif" onMouseOver = "A5()">

 </body>
</html>
```

Aufgabe3

So könnte der Quelltext der zweiten Seite aussehen:

```
<html>
 <head>
  <script>
     function wiedersehen()  {
       alert("auf Wiedersehen");
     }
```

```
  </script>
 </head>

 <body onBeforeUnload = "wiedersehen()">
 </body>
</html>
```

Aufgabe 4

```
<body onKeyPress = "alert('Lass die Finger von der
Tastatur!')">
```

Befehle zur Textverarbeitung

Die Länge eines Wortes bzw. Textes wird durch das Attribut *length* angegeben. Beispiel: **`n = wort.length`**

```
<script>
  function willkommen()  {
    var wort1="Gymnasium", wort2, n, ch;
    alert(wort1.length);   //liefert die Zahl 9
    if (wort2 == "Inf")  alert("Inf");
    wort2 = wort1.substring(2, 3);   //liefert den Buchstaben m
    alert(wort2);
    wort3 = wort1.substr(2, 3);   //liefert die Buchstaben mna
    alert(wort3);
    n = wort1.indexOf("nasi");   //liefert den Wert 3
    alert(n);
    wort4 = wort1.slice(2);   //liefert den Text mnasium
    alert(wort4);
    wort4 = wort1 + wort2;
    alert(wort4);
    wort5 = wort1.toUpperCase();   //liefert das Wort GYMNASIUM
    alert(wort5);
    wort6 = wort1.toLowerCase();   //liefert das Wort gymnasium
    alert(wort6);
    ch = wort1.charAt(3);   //liefert das Zeichen n
    alert(ch);
    wort7 = "heute ist es schön und es ist kalt";
    wort7 = wort7.replace("ist","wird");
    alert(wort7);
  }
</script>
```

Beachte, dass die Variable *wort2* unbedingt deklariert werden muss!

Die Methode ***substr****(startposition, laenge)* liefert im obigen Beispiel die drei Buchstaben "mna" zurück: *startposition* ist der erste zu kopierende Buchstabe, wobei *JavaScript* bei Null anfängt zu zählen. *laenge* gibt die Anzahl der zu kopierenden Buchstaben an.

Die Methode ***substring****(anfang, ende)* liefert im obigen Beispiel hingegen nur den Buchstaben "m" zurück: *anfang* ist der erste zu kopierende Buchstabe, wobei JavaScript bei Null anfängt zu zählen, *ende* ist der letzte Buchstabe, der allerdings nicht mehr kopiert wird.

Völlig analog wie ***substring****(anfang, ende)* funktioniert auch die Methode ***slice****(anfang, ende).* Allerdings können hier einer oder beide Parameter auch

negativ sein. In dem Fall wird vom Ende des Wortes an rückwärts gezählt. Beispiel: ***wort4 = wort1(-3,-1)*** kopiert aus dem Wort *wort1* den drittletzten und den vorletzten Buchstaben.

Der Befehl ***wort4=wort1.slice(2);*** kopiert aus dem Wort *wort1* einen Teilstring ab der 2. Stelle (Beginn bei 0) bis zum Ende heraus und das Ergebnis wird dem Wort *wort4* zugeordnet.

Der Befehl ***n=wort1.indexOf("nasi")*** überprüft, ob das Teilwort *"nasi"* in dem Wort *wort1* enthalten ist. Falls ja, wird die Stelle, an der das Teilwort (erstmalig) beginnt, der Variablen ***n*** zugewiesen. Beachte, dass wie gewöhnlich ab 0 gezählt wird. Im obigen Beispiel erhält die Variable ***n*** also den Wert 3. Falls das Teilwort nicht enthalten ist, wird der Wert ***-1*** zurückgeliefert.

Der Befehl ***ch=wort1.charAt(3)*** ermittelt das Zeichen an der dritten Stelle (Beginn bei 0). In diesem Beispiel also den Buchstaben *n*.

Die Methoden *toUpperCase* und *toLowerCase* erklären sich von selbst. **Achte hierbei exakt auf Groß-Kleinschreibung !**

Die Methode ***replace****(originalTeilstring, ersatzteilstring)* liefert als Ergebnis einen String, in dem **nur einmal** ersetzt wurde. An dem *originalTeilstring* selbst wird dabei nichts geändert.
Beispiel:
```
wort = "abcabc";
wort2 = wort.replace("b", "z");
alert(wort)   // liefert "abcabc"
alert(wort2)  // liefert "azcabc"
```

Gegebenenfalls muss man diese Methode öfter anwenden. **Mit dieser Methode kann man auch Teilstrings löschen**, indem man sie durch Leerstrings ersetzt.

Das folgende Beispiel testet, ob ein eingegebener Name nur erlaubte Zeichen enthält. Beachte, dass in diesem Beispiel auch der Gedankenstrich (bei Doppelnamen) und das Leerzeichen zu den erlaubten Zeichen gehört!

```
function teste()  {
  var ch;
  var Zeichenmenge =
    "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZäöüßÄÖÜ- ";
  var eingabe = prompt("Bitte geben Sie Ihren Namen ein!");
  for (var i=0; i < eingabe.length; i++)  {
     ch = eingabe.charAt(i);
     if (Zeichenmenge.indexOf(ch) == -1)
        alert("dies ist kein Name");
  }
}
```

Alle Schriftzeichen sind auf der Basis des *Unicode-Zeichensatzes* nummeriert. Die ersten 128 Zeichen entsprechen dem sog. *ASCII-Code* (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange).
Die Großbuchstaben beginnen bei 65 (entsprechend „A“) und enden bei 90 (entsprechend „Z“). Die zugehörigen Kleinbuchstaben liegen zwischen 97 („a“) und 122 („z“); die Ziffern liegen zwischen 48 (entsprechend „0“) und 57 („9“).

Im Folgenden ist eine kleine Code-Tabelle angegeben für einige mathematische Sonderzeichen:

≤	8804
≥	8805
≠	8800
≈	8776
±	177
∞	8734

```
<html>
 <head>
  <script>
   function zeichensatz()  {
     var wort = "Informatik";
     var n = wort.charCodeAt(0);   // n=73
     var m = wort.charCodeAt(1);   // m=110
     alert(n);
     alert(m);
     var neuesWort = String.fromCharCode(65, 66, 67);
     // es handelt sich hier um eine sog. Klassenmethode der Klasse String!
     alert(neuesWort);    // es wird „ABC“ ausgegeben.
   }
  </script>
 </head>

 <body onLoad = "zeichensatz()">
 </body>
</html>
```

Problematik Text – Zahl

Java-Script wandelt immer dann, wenn es „denkt“, es wäre nötig, einen String (=Text) in eine Zahl um und umgekehrt. Das vereinfacht manches, allerdings nicht alles.

Problematisch ist dabei auch, dass das Plus-Zeichen unterschiedliche Bedeutungen haben kann:
Steht es zwischen zwei Zahlen, so bedeutet es Addition der beiden Zahlen.
Steht es zwischen zwei Strings, so bedeutet es Verkettung der Strings.
Steht es zwischen einem String und einer Zahl, so wird zuerst die Zahl in einen String umgewandelt und danach werden die beiden Strings verkettet.
Steht es in Anführungszeichen, also innerhalb eines Strings, so sieht es einfach nur aus wie ein Kreuz und hat keine weitere Bedeutung.

Im Folgenden werden einige Beispiele gegeben:

```
var a = "12";
var x = 5;
var y = 10;
alert(x + y);      // Ausgabe: 15
alert(a + x);      // Ausgabe: 125
alert("a + x");    // Ausgabe: a + x
alert("x + y = " + x + y)       // Ausgabe: x + y = 510
alert("x + y = " + (x + y))     // Ausgabe: x + y = 15
alert("x" + "y" + " = " + (x + y))    // Ausgabe: xy = 15
alert(x+ " + " + y + " = " + (x + y))// Ausgabe: 5 + 10 = 15
```

Wenn man eine Zahl und keinen Text haben will, sollte man möglichst selbst schon eine Umwandlung vornehmen:

```
wort = "12.84";
x = parseFloat(wort);  // bewirkt, dass  x=12.84
y = parseInt(wort); // bewirkt, dass  y=12 also der ganzzahlige Anteil
```

Eine andere mögliche, allerdings nicht empfehlenswerte Umwandlungsmethode (von Text zu Zahl) ist ***eval()***
Beispiel:

```
a = "5"; b = "10";
c = a+b;  würde ergeben  c = "510"
c = eval(a+b);  ergibt  c = 510
c = eval(a) + eval(b);  ergibt  c = 15
```

Was ist daran nicht empfehlenswert?
Die Methode **eval** interpretiert den übergebenen String als Java-Script-Code und führt ihn aus. Das ist manchmal durchaus sehr interessant, manchmal aber auch gefährlich. Betrachte folgendes Beispiel:

```
var eingabe = prompt("Gib Zahl ein", "");
var x = eval(eingabe);
```

Falls der Besucher keine Zahl eingeben sollte, sondern stattdessen die Anweisung `confirm("Hallo Welt");` so wird eben diese Anweisung ausgeführt (welche ja noch relativ harmlos ist).

Ein böswilliger Besucher könnte aber auch ein komplettes Java-Script-Programm eingeben, welches dann entsprechend Schaden anrichten könnte.

Man sollte bei der Abfrage von Benutzereingaben also niemals die Methode **eval** benutzen!

Also: Falls die Variable n eine ganze Zahl sein soll, dann möglichst nicht
`n = prompt("bitte eine Zahl eingeben","0");` sondern
`n = parseInt(prompt("bitte eine Zahl eingeben","0"));`

Aufgaben

1. Ein Passwort soll eingegeben werden. Das richtige Passwort lautet „Informatik ist toll“. Dabei soll es nicht auf Groß- oder Kleinschreibung ankommen, d.h. auch die Eingabe „INforMAtIK isT toLL“ wird akzeptiert. Ausgegeben wird entweder „OK“ oder „falsches Passwort!“

2. Im Zahlungsverkehr im Bankwesen muss man sehr oft längere Zahlenfolgen eingeben, beispielsweise die sog. IBAN (englisch: ***International Bank Account Number***, deutsch: Internationale Bankkontonummer). Eine Beispiel-IBAN wäre etwa DE34441524900001234583
Diese IBAN kann und wird auch oft mit beliebig vielen Leerzeichen eingegeben. Das macht die Nummer (für Menschen) besser lesbar. Man könnte also DE34 4415 2490 0001 2345 83 eingeben oder auch
DE 34 441 524 90 0001 2345 83
Wenn der Rechner diese Nummer weiterverarbeiten will, muss er erst einmal sämtliche Leerzeichen entfernen. Ansonsten ließe sich kein Vergleich durchführen.

Der Benutzer soll obige Beispiels-IBAN eingeben (mit beliebig vielen Leerzeichen an beliebig vielen Stellen). Der Rechner entfernt alle Leerzeichen aus der Eingabe und kontrolliert dann, ob es sich um die richtige IBAN handelt. Eine entsprechende Meldung wird ausgegeben.

3. Ein Wort wird eingegeben. Es soll untersucht werden, ob es ein Palindrom ist.

4. Eine Telefonnummer wird eingegeben. Falls diese mit der Ziffernfolge „0231“ beginnt, erfolgt die Begrüßung *„Hallo Dortmund“*.

5. Ein längerer Satz wird eingegeben, welcher mehrfach das Wort „*ist*“ enthält. Jedes Vorkommen dieses Teilwortes soll in dem Satz gelöscht werden.

6. Ein Wort wird eingegeben. Der Rechner gibt die Länge des Wortes aus.

7. Ein Text wird eingegeben. Der Rechner gibt aus, wie oft der Buchstabe ‚e' in diesem Text vorkommt.
Bemerkung: die relative Häufigkeit der einzelnen Buchstaben hängt von der Sprache (deutsch, englisch usw.) ab. Man kann also bei einem längeren Text die Sprache aufgrund der Buchstabenhäufigkeit identifizieren.

8. Eine Telefonnummer wird eingegeben. Der Rechner untersucht, ob alle eingegebenen Zeichen in der Menge „0123456789-/" enthalten sind.

9. Der Anwender gibt einen Buchstaben ein (mit der Methode *prompt*) und der Computer gibt die zugehörige ASCII-Nummer aus.

10. Der Anwender gibt die ASCII-Nummer eines Zeichens ein und der Computer gibt das zugehörige Zeichen aus.

11. Cäsar-Verschlüsselung.
Der Anwender gibt ein Wort (nur Großbuchstaben) ein. Jeder Buchstabe wird nun durch seinen dritten Nachfolger im Alphabet ersetzt (der dritte Nachfolger von Z ist C). Das kodierte Wort wird ausgegeben.
Beispiel: Eingabe = „ANTONZ", Ausgabe = „DQWRQC"

12. Der Anwender gibt einen ganzen Satz (mit Groß- und Kleinbuchstaben, Ziffern und Satzzeichen) ein. Nur die Buchstaben werden durch ihre dritten Nachfolger im Alphabet ersetzt. Der kodierte Satz wird ausgegeben.

13. Wie Aufgabe 12. Statt des dritten Nachfolgers soll der n-te Nachfolger benutzt werden, wobei n eine beliebige Verschiebung ist, welche der Benutzer eingeben kann.

14. Ein nach Aufgabe 11 bis 13 kodiertes Wort oder Satz wird eingegeben und der Computer soll es wieder dekodiert ausgeben.

Die folgende Aufgabe ist etwas schwieriger:
15. Ein längerer Satz wird eingegeben. Der Satz soll so bearbeitet werden, dass bei allen darin vorkommenden Worten der jeweilige Anfangsbuchstabe groß geschrieben wird. Hinweis: die einzelnen Worte sollen durch Leerzeichen voneinander getrennt sein.

Lösungen

Aufgabe1

```
<script>
 function f()   {
   var eingabe = prompt("Bitte gib das Passwort ein");
   eingabe = eingabe.toUpperCase();
   if (eingabe == "INFORMATIK IST TOLL")  alert("OK");
   else alert("falsches Passwort!");
 }
</script>
```

Aufgabe2 (Möglichkeit 1)

```
<script>
 function f()   {
   var iban = prompt("Bitte gib die IBAN ein!");
   var kopie = "";
   for (var i=0; i<iban.length; i++)
     if (iban.substr(i, 1)!=" ") kopie = kopie + iban.substr(i, 1);
   if (kopie == "DE34441524900001234583") alert("OK");
   else alert("falsche IBAN");
 }
</script>
```

Aufgabe2 (Möglichkeit 2)

```
<script>
 function f()   {
   var iban = prompt("Bitte gib die IBAN ein!");
   var kopie = iban;
   while(kopie.indexOf(" ") != -1)
       kopie = kopie.replace(" ","");
   if (kopie == "DE34441524900001234583") alert("OK");
   else alert("falsche IBAN");
 }
</script>
```

Aufgabe3

```
<script>
  function pal()  {
    var wort = prompt("Bitte ein Wort eingeben");
    var laenge = wort.length;
    var kopie = "", i;
    for (var i=0; i<laenge; i++)
      kopie = wort.substr(i, 1) + kopie;
    if (kopie == wort) alert("es ist ein Palindrom");
    else alert("es ist kein Palindrom");
  }
</script>
```

Aufgabe4

```
<script>
  function f()   {
    var telefon= prompt("Bitte gib deine Telefonnummer ein!");
    if (telefon.indexOf("0231") == 0) alert("Hallo Dortmund");
  }
</script>
```

Aufgabe5

```
<script>
  function f()   {
    var satz = prompt("Bitte gib einen Satz ein,\nder u.a.
                                    das Wort ist enthält!");
    while(satz != satz.replace("ist","") )
    satz = satz.replace("ist","");
    alert(satz);
  }
</script>
```

Aufgabe6

```
<script>
   var wort = prompt("Bitte ein Wort eingeben!","");
   var n = wort.length;
   alert("Das Wort hat die Laenge " + n);
</script>
```

Aufgabe7 (Möglichkeit 1)

```
<script>
   var wort = prompt("Bitte einen Text eingeben!","");
   var zeichen = "";
   var anzahl = 0;
   for (var i=0; i< wort.length; i++)  {
      zeichen = wort.charAt(i);
      if (zeichen == "e") anzahl++;
   // möglich wäre auch: if (wort.substr(i,1) == "e") anzahl++;
   }
   alert(anzahl);
</script>
```

Aufgabe7 (Möglichkeit 2)

```
<script>
   var wort = prompt("Bitte einen Text eingeben!","");
   var position;
   var anzahl = 0;
   do {
     position = wort.indexOf("e");
     if (position >= 0) {
       anzahl++;
       wort = wort.replace("e", "");
     }
   }
   while (position >= 0);
   alert("Anzahl e: " + anzahl);
</script>
```

Aufgabe8

```
<script>
   var ch;
   var zeichenmenge = "0123456789-/";
   var telefon = prompt("gib Telefonnummer ein!");
   for (var n = 0; n < telefon.length; n++)   {
     ch = telefon.charAt(n);
     if (zeichenmenge.indexOf(ch) == -1)   {
       alert("keine gültige Telefonnummer!");
       break;
     }
   }
</script>
```

Aufgabe9

```
<script>
   var wort = prompt("Bitte einen Buchstaben eingeben!","");
   var n = wort.charCodeAt(0);
   alert("der ASCII-Code ist " + n);
</script>
```

Aufgabe10

```
<script>
   var n;
   n = parseInt(prompt("Bitte ASCII-Code-Nummer eingeben!",""));
   var zeichen = String.fromCharCode(n);
   alert("das zugehoerige Zeichen ist " + zeichen);
</script>
```

Aufgabe11

```
<script>
   var wort = prompt("Bitte Wort in Grossbuchstaben eingeben!","");
   var zeichen="", neuwort = "";
   var code;
   for (var i=0; i< wort.length; i++)  {
      code = (wort.charCodeAt(i) - 65 +3) % 26 + 65;
      zeichen = String.fromCharCode(code);
      neuwort = neuwort + zeichen;
   }
   alert(neuwort);
</script>
```

Aufgabe12

```
<script>
   var satz = prompt("Bitte einen Satz eingeben!","");
   var zeichen="";
   var code;
   var neusatz = "";
   for (var i=0; i< satz.length; i++)  {
      code = satz.charCodeAt(i);
      if (65<= code && code <= 90)
         code = (code - 65 +3) % 26 +65;
      else if (97 <= code && code <= 122)
         code = (code - 97 +3) % 26 + 97;
      zeichen = String.fromCharCode(code);
      neusatz = neusatz + zeichen;
   }
   alert(neusatz);
</script>
```

Aufgabe13

```
<script>
   var satz = prompt("Bitte einen Satz eingeben!","");
   var n = parseInt(prompt("Verschiebungszahl n eingeben", "0"));
   var zeichen="";
   var code;
   var neusatz = "";
   for (var i=0; i< satz.length; i++)  {
     code = satz.charCodeAt(i);
     if (65<= code && code <= 90)
        code = (code - 65 + n) % 26 + 65;
     else if (97 <= code && code <= 122)
        code = (code - 97 + n) % 26 + 97;

     zeichen = String.fromCharCode(code);
     neusatz = neusatz + zeichen;
   }
   alert(neusatz);
</script>
```

Zeitverzögerte Methoden

Die Methode **`setTimeout("Anweisungen", AnzahlMillisekunden)`** erlaubt es, JavaScript-Befehle nach Ablauf einer bestimmten Zeit einmalig auszuführen. Die Anweisungen (üblicherweise ist das nur eine einzige Funktion) müssen dabei in doppelten Anführungszeichen stehen.

```
var x=4;
setTimeout("x=8; alert(x); t();", 3000);
alert("Meldung 1");
alert("Meldung 2");
alert("Meldung 3");
alert("Meldung 4");

function t()  {
  alert("Hallo Welt!");
}
```

JavaScript merkt sich, wie viele Millisekunden nach Lesen der Methode *setTimeout()* diese ausgeführt werden soll. *JavaScript* wartet nicht auf die Ausführung sondern macht sofort mit den nächsten Befehlen weiter und fügt die entsprechende Anweisung zur vorgegebenen Zeit ein.
Beachte allerdings im obigen Beispiel, dass alle *alert*-Anweisungen solange auf dem Bildschirm stehen bis sie mit OK bestätigt werden! Im Beispiel werden alle 4 Meldungen zuerst ausgegeben, egal wie lange sie auf dem Bildschirm stehen, bevor die Anweisungen im *setTimeOut*-Befehl ausgeführt werden. Das ist etwas unverständlich!

Sehr oft wird diese Methode *setTimeout()* so benutzt wie oben dargestellt. Allerdings liefert diese Methode *setTimeout()* auch eine Identifikationsnummer (ID) des dabei benutzten *Timers* zurück. Mit Hilfe dieser ID und der Methode *clearTimeout(ID)* kann man den *Timer* auch vor der eigentlichen Ausführung noch wieder stoppen.

```
var n;
n = setTimeout("t()", 6000);
var eingabe = confirm("Soll der Timer gestoppt werden?");
if (eingabe == true) clearTimeout(n);

function t()  {
  alert("Hallo Welt!");
}
```

Völlig analog funktioniert die Methode
setInterval("Anweisungen", AnzahlMillisekunden)

Allerdings werden hier die Anweisungen (meist nur eine einzige Funktion) immer wieder nach der angegebenen Zeit wiederholt.

```
<html>
 <head>
  <script>
    var i = 0;

    function zaehlHoch()   {
      i++;
      document.Formular1.fenster.value = i;
    }

    setInterval("zaehlHoch()", 1000);
  </script>
 </head>


 <body>
   <form name="Formular1">
     <input name="fenster" type="text" size="2"
        style="font-size:20pt; color:red;
        background-color: green; font-weight: bold;">
   </form>
 </body>
</html>
```

Auch die Methode *setInterval()* liefert eine Identifikationsnummer zurück, mit deren Hilfe man die immer wiederkehrende Ausführung stoppen kann.

```
<html>
 <head>
  <script>
   var i = 0;
   var n = 0;

   function zaehlHoch()   {
     i++;
     document.Formular1.fenster.value = i;
   }

   function stopp() {
     clearInterval(n);
   }

   n = setInterval("zaehlHoch()",1000);
  </script>
 </head>


 <body>
   <form name="Formular1">
     <input name="fenster" type="text" size="2"
        style="font-size:20pt; color:red;
        background-color: green; font-weight: bold;">
   </form>
   <br />

   <a href="javascript: stopp();"> Zaehler stoppen </a>

 </body>
</html>
```

Laufschriften

```
<html>
 <head>
  <script>

   var satz="Informatik ist ein tolles Fach!  ";
   var laenge = satz.length;

   function lauf()   {
     var ch = satz.charAt(0);
     satz = satz.substr(1, laenge - 1) + ch;
     document.Formular1.fenster.value=satz;
     setTimeout("lauf()",200);     // rekursiver Aufruf
   }

   setTimeout("lauf()",75);
  </script>
 </head>


 <body>
   <form name="Formular1">
     <input name="fenster" type="text" size="30"
        style="font-size:20pt; color:red;
        background-color: green; font-weight: bold;">
   </form>
 </body>
</html>
```

Aufgaben

1. Ändere obiges Programm derart, dass die Laufschrift langsamer läuft!

2. Ändere den Algorithmus aus Aufgabe 1 so, dass eine nach rechts laufende Laufschrift entsteht!

3. Es sollen gleichzeitig zwei Laufschriften erzeugt werden: Eine Schrift läuft nach links, die andere nach rechts. Löse diese Aufgabe in zwei Versionen:
a) mit zwei Formularen
b) mit einem einzigen Formular (allerdings mit zwei Eingabe-Fenstern)!

4. Löse obige Aufgaben auch mit der Anweisung *setInterval()* ! Achte aber darauf, dass dieser Befehl nicht innerhalb einer rekursiven Funktion aufgerufen wird!

Lösung der Aufgabe 2:
Ändern muss man nur die ersten beiden Anweisungen in der Funktion lauf()

```
   function lauf()   {
     var ch = satz.charAt(laenge - 1);
     satz = ch + satz.substr(0, laenge - 1);
     document.Formular1.fenster.value=satz;
     setTimeout("lauf()",200);     // rekursiver Aufruf
   }
```

Lösung der Aufgabe 3a):

```
<script>
   var message = "Informatik ist super !!!                ";
   var ausgabe = message;


 function laufschrift()  {
   var zeichen = message.substr(message.length -1, 1);
   message =zeichen+message.substring(0,message.length-1);
   document.Banner.Bannertext.value = message;
   window.setTimeout("laufschrift()", 100);
 }


  function laufschrift2()  {
   var zeichen = ausgabe.substr(0,1);
   ausgabe = ausgabe.substring(1,ausgabe.length)+ zeichen;
   document.Banner2.Bannertext.value = ausgabe;
   window.setTimeout("laufschrift2()", 100);
 }

</script>
</head>

<body onLoad="laufschrift(); laufschrift2()">
 <center>
  <form name = "Banner">
    <input size = 40  style = "font-size: 30pt; color:red;
      background-color: green; font-weight: bold;"
      name = "Bannertext">
    </input>
  </form>
  <br />
```

```
  <form name = "Banner2">
    <input size = 40  style = "font-size:30pt; color:blue;
      background-color: yellow; font-weight: bold;"
      name = "Bannertext">
    </input>
  </form>

 </center>
</body>
</html>
```

Lösung der Aufgabe 3b):

```
<script>
   var message = "Informatik ist super !!!              ";
   var ausgabe = message;

 function laufschrift()  {
   var zeichen = message.substr(message.length - 1, 1);
   message =zeichen+message.substring(0,message.length-1);
   document.Banner.Bannertext.value = message;
   window.setTimeout("laufschrift()", 100);
 }


  function laufschrift2()  {
   var zeichen = ausgabe.substr(0, 1);
   ausgabe = ausgabe.substring(1,ausgabe.length)+ zeichen;
   document.Banner.Bannertext2.value = ausgabe;
   window.setTimeout("laufschrift2()", 300);
 }

</script>
</head>

<body onLoad="laufschrift(); laufschrift2()">
 <center>
  <form name = "Banner">
    <input size = 40  style = "font-size: 30pt; color:red;
      background-color: green; font-weight: bold;"
      name = "Bannertext">
    </input>
    <input size = 40  style = "font-size:30pt; color:blue;
      background-color: yellow; font-weight: bold;"
      name = "Bannertext2">
    </input>
```

```
  </form>
  <br />
 </center>
</body>
</html>
```

Auslagern von Java-Script-Code

Java-Script-Code läßt sich auslagern in eine separate *Text*-Datei. In der HTML-Seite muss nur das *Script*-Tag entsprechend modifiziert werden:

```
<script src="meinGeheimerJavaCode">
</script>
```

Die entsprechende Textdatei darf nur den reinen *Java-Script*-Code enthalten, also insbesondere kein *Script-Tag* mehr. Achte bei der Wahl des Dateinamens auch darauf, ob ein Anhängsel im Namen existiert. Der Name **`meinGeheimerJavaCode`** ist nicht derselbe wie **`meinGeheimerJavaCode.txt`**

Hinweis: Oft werden im Windows-Explorer sog. bekannte Anhängsel wie etwa *com, exe, txt, jpg* und viele mehr nicht angezeigt. Um diese Anhängsel sichtbar zu machen, muss man die Eigenschaften des entsprechenden Window-Ordners anpassen.

Der Aufruf einer Java-Funktion erfolgt wie gewohnt im HTML-Text.

Mit jedem Browser lässt sich der Quelltext (bestehend aus HTML- und *Java-Script*-Text) einer Webseite anzeigen. Die Browser *Internet-Explorer* und *Firefox* zeigen (im Jahr 2018) den Quelltext der ausgelagerten Code-Datei nicht an.
Allerdings muss diese Datei natürlich beim Aufruf der HTML-Seite auch automatisch in den Rechner geladen und dort irgendwo gespeichert werden. Sie befindet sich (allerdings unter anderem Namen) im sog. Cache-Speicher des Rechners bzw. Browsers und kann gegebenenfalls dort nachgelesen werden.

Zum Verstecken von Passwörtern eignet sich diese Auslagerungsmethode also nicht wirklich. Sie ist nur sinnvoll, wenn mehrere Programme teilweise denselben Quellcode benutzen.

Aufgabe:
Benutze irgendein von dir erstelltes, funktionierendes *Java-Script*-Programm und lagere den *Java-Script*-Code, wie oben beschrieben, in eine Datei aus.

Die *switch-case*-Anweisung

Mit *switch* leitet man eine Fallunterscheidung ein (*switch = Schalter*). Dahinter folgt, in runde Klammern eingeschlossen, eine Variable oder ein Ausdruck, für dessen aktuellen Wert man die Fallunterscheidung durchführt. Im nachfolgenden Beispiel ist das die Variable mit dem Namen *note*.

Die einzelnen Fälle, die man abfragen möchte, werden innerhalb eines einzigen geschweiften Klammernpaares notiert. Jeden einzelnen Fall leitet man mit *case* ein (*case = Fall*). Dahinter folgt die Angabe des Wertes, auf den geprüft wird.

Wichtig ist dabei auch das Wort *break* am Ende jedes Falles. Wenn man dieses Wort weglässt, werden nämlich alle nachfolgenden Fälle auch ausgeführt.

Für den Fall, dass keiner der definierten Fälle zutrifft, kann man am Ende der Fallunterscheidung den Fall *default :* definieren. Die dahinter stehenden Anweisungen werden ausgeführt, wenn keiner der anderen Fälle zutrifft. Hinter der letzten Anweisung benötigt man keinen *break*-Befehl mehr.

```
<script>
  function abfrage()  {
    var note;
    note = prompt("Notenziffer eingeben", "");
    switch(note)   {
      case "1": alert("sehr gut");
                break;
      case "2": alert("gut");
                break;
      default:  alert("Naja !");
    } // end of switch
  }

 </script>
</head>

<body onLoad = "abfrage();">
.....
```

Aufgaben

Löse die folgenden Aufgaben mit der switch-case-Anweisung!

1. Der Besucher der Homepage soll seinen Namen angeben. Die vier Freunde Willi, Peter, Katja und Doris werden herzlicher begrüßt als alle anderen Besucher.

2. Der Besucher gibt den Namen seines Lieblingsunterrichtsfaches ein. Der Rechner gibt einen entsprechenden Kommentar (toll, super, ….) aus.

3. Es wird ein Notenwort (sehr gut, ….) eingegeben. Der Rechner gibt die zugehörige Notenziffer aus. Er meldet auch, wenn ein Notenwort falsch geschrieben sein sollte.

Die Klasse *Math*

Die Klasse *Math* besitzt sog. Klassenmethoden, d.h. man muss kein Objekt dieser Klasse erzeugen, um diese Methoden benutzen zu können.

Beispiel: eine mögliche Zuweisung lautet: **`x = Math.sqrt(0.16)`**

```
abs(-3.12)

acos(0.1)        // Ergebnisse in Radiant
asin(0.34)
atan(1.3)

cos(1.2)         // Die Winkel müssen in Radiant angegeben sein.
sin(2.4)
tan(0.5)

ceil(3.4)        // ermittelt die nächsthöhere ganze Zahl

exp(3)           // ermittelt die Potenz  e^3

floor(3.4)       // ermittelt die nächstniedrigere ganze Zahl

log(5.3)         // ermittelt den natürlichen Logarithmus  ln 5.3

max(1.2, 3.7)
min(1.2, 5.6)

pow(7.3, 2.1)    //  ermittelt die Potenz  7.3^2.1

random()         // ermittelt eine Zufallszahl zwischen
                 // (einschließlich) 0 und (ausschließlich) 1

round(4.76)      // ermittelt die mathematisch gerundete ganze Zahl

sqrt(3.4)        // ermittelt die Quadratwurzel

PI               // Kreiszahl π
E                // Eulersche Zahl  e
```

Beispiele

Eine **rationale** Zufallszahl z zwischen (einschließlich) 0 und (ausschließlich) 12 wird erzeugt durch die Anweisung `z = 12*Math.random();`

Eine **ganzzahlige** Zufallszahl z zwischen (einschließlich) 0 und (einschließlich) 12 wird erzeugt durch die Anweisung

```
z = Math.round(12* Math.random());
```

Bemerkung: Aufgrund der Rundung ist die Wahrscheinlichkeit, eine der beiden Randwerte zu erzeugen, deutlich geringer. Erkläre dies! Durch welche andere Anweisung ließe sich dies verhindern?

Eine **rationale** Zufallszahl z zwischen (einschließlich) 5 und (ausschließlich) 12 wird erzeugt durch die Anweisung `z = 5 + 7*Math.random();`

Eine **ganzzahlige** Zufallszahl z zwischen (einschließlich) 5 und (einschließlich) 12 wird erzeugt durch die Anweisung

```
z = 5 + Math.round(7*Math.random());
```

Wenn ein Kreis den Radius r=4 haben sollte, so berechnet man seine Fläche mit

```
z = Math.PI * 4 * 4;
```

Aufgaben

1. Der Besucher der Homepage gibt den Durchmesser eines Kreises an. Der Computer gibt daraufhin den Umfang und Flächeninhalt des Kreises aus.

2. Der Computer soll 30 Mal würfeln, also Zufallszahlen von 1 bis 6 ausgeben. Alle 30 Zufallszahlen sollen in einem einzigen Ausgabefenster erscheinen.

3. Der Computer soll 1 000 Mal würfeln, aber diese Zufallszahlen nicht ausgeben. Stattdessen soll er zählen, wie oft jede einzelne Zahl gewürfelt worden ist. Anschließend wird das Ergebnis der absoluten und der relativen Häufigkeiten ausgegeben.

4. Der Computer soll 30 Mal mit zwei Würfeln würfeln und die Augensumme beider Würfel ausgeben. Alle 30 Summen sollen in einem einzigen Ausgabefenster erscheinen.

5. Der Computer soll 1 000 Mal mit zwei Würfeln würfeln und zählen, wie oft jede mögliche Augensumme (von 2 bis 12) vorkam. Anschließend wird das Ergebnis der relativen Häufigkeiten ausgegeben. Welche Augensumme kommt am häufigsten vor?

6. Dem Besucher wird eine simple Additionsaufgabe gestellt (mit natürlichen Zufallszahlen zwischen 0 und 20). Die Eingabe seines Ergebnisses wird vom Computer mit „richtig“ oder „falsch“ kommentiert.

7. Dem Besucher wird eine simple Subtraktionsaufgabe gestellt (mit natürlichen Zufallszahlen zwischen 0 und 20). Sorge dafür, dass das Ergebnis auf keinen Fall negativ sein wird! Die Eingabe seines Ergebnisses wird vom Computer entsprechend kommentiert.

8. Der Besucher gibt 3 Noten (als Ziffern) ein. Der Computer berechnet den Durchschnittswert und gibt ihn auf eine Stelle hinter dem Komma genau aus. Hinweise: Beachte, dass z.B. bei der Summenbildung der drei Noten 2, 3, 2, der Wert 7 herauskommt und nicht das Wort „232“! Multipliziere den Durchschnittswert mit 10, runde das Ergebnis auf eine ganze Zahl und dividiere diese anschließend durch 10.

9. Dem Besucher werden fünf einfache, zufällige Multiplikationsaufgaben gestellt (die Faktoren sollen ganzzahlig zwischen ausschließlich 0 und einschließlich 10 liegen). Die Anzahl der richtig eingegebenen Lösungen wird mitgezählt.
Anschließend wird dem Besucher eine Note für seine Leistung mitgeteilt.

10. Wie Aufgabe 9, nur diesmal sollen es auch zufällige Rechenoperatoren sein.

11. Pythagoras. Der Besucher wird nacheinander aufgefordert, die Längen der Dreiecksseiten a, b und c einzugeben (natürliche Zahlen!). Der Computer teilt ihm mit, ob dieses Dreieck rechtwinklig ist. Hinweis: Es ist nicht bekannt, welche Seite Kathete oder Hypotenuse ist.

12. Der Besucher gibt einen Winkel in Grad ein und der Computer gibt denselben Winkel in Radiant aus.

13. Der Besucher gibt einen Winkel in Radiant ein und der Computer gibt denselben Winkel in Grad aus.

14. Dreiecksberechnung. Der Besucher wird nacheinander aufgefordert, die Längen der Dreiecksseiten a, b und c einzugeben. Der Computer teilt ihm zunächst mit, ob ein solches Dreieck überhaupt möglich ist (Summe zweier Seiten muss immer größer als die dritte Seite sein!). Danach berechnet der Computer die drei Winkel α, β und γ und gibt sie (in Grad) aus.

15. Lottozahlen. Der Computer soll sechs zufällige Zahlen zwischen 1 und 49 ausgeben (einschließlich der Grenzen).
a) leichte Version: Die Zahlen dürfen teilweise identisch sein.
b) realistische Version: Es müssen sechs unterschiedliche Zahlen sein.

16. Geburtstagsrechnung. Ermittle, wie viele Schüler in einer Klasse sein müssen, damit die Wahrscheinlichkeit, dass mindestens 2 Schüler am selben Tag ihren Geburtstag feiern, größer als 50% ist!
Lösungsweg: Berechne zunächst die Gegenwahrscheinlichkeit! Anleitung: Die Wahrscheinlichkeit, dass der 2. Schüler nicht am selben Tag wie der 1. Schüler Geburtstag hat, beträgt $\frac{364}{365}$. Die Wahrscheinlichkeit, dass auch der 3. Schüler nicht mit einem der ersten beiden gemeinsam Geburtstag hat, beträgt dann $\frac{364}{365} \cdot \frac{363}{365}$. Die Wahrscheinlichkeit, dass auch der 4. Schüler nicht mit einem der ersten drei gemeinsam Geburtstag hat, beträgt dann $\frac{364}{365} \cdot \frac{363}{365} \cdot \frac{362}{365}$. Die Wahrscheinlichkeit, dass auch der n. Schüler nicht mit einem der ersten (n-1) gemeinsam Geburtstag hat, beträgt dann $\frac{364}{365} \cdot \frac{363}{365} \cdot \frac{362}{365} \cdot \ldots \cdot \frac{366-n}{365}$.

Erweitere dein Programm so, dass praktisch eine Tabelle ausgegeben wird, welche die Schülerzahlen von 2 bis 30 zusammen mit den zugehörigen Wahrscheinlichkeiten enthält!

Lösungen

Aufgabe 1

```
<script>
   var d, r, U, A, c;
   d = prompt("Bitte Durchmesser eingeben!", "");
   r = d/2;
   U = 2*Math.PI*r;
   U = Math.round(100*U)/100;    // auf zwei Stellen runden
   A = Math.PI*r*r;
   A = Math.round(100*A)/100;
   c = String.fromCharCode(8776);  // das Zeichen ≈
   alert("U " +c + U + " und A " + c + A);
</script>
```

Aufgabe 2

```
   function wuerfel30Mal()  {
       var z;
       var ausgabe = "";
       for (var i = 1; i <= 30; i++)  {
          z = Math.ceil(6*Math.random());
          ausgabe = ausgabe + z + "\n"
       }
       alert(ausgabe);
   }
```

Aufgabe 3

```
  function wuerfel()  {
     var eins = 0;
     var zwei = 0;
     var drei = 0;
     var vier = 0;
     var fuenf = 0;
     var sechs = 0;
     var z;

     for (var i = 1; i <= 1000; i++)  {
      /* z = Math.round(1 + 5*Math.random());
         obiger Befehl würde leider falsche Ergebnisse für die Augenzahlen 1
```

```
        und 6 liefern. Erkläre dies! */
        z = Math.ceil(6*Math.random());
      switch(z)   {
        case 1: eins++; break;
        case 2: zwei++; break;
        case 3: drei++; break;
        case 4: vier++; break;
        case 5: fuenf++; break;
        case 6: sechs++;
      } // end of switch
    } // end of for

    var ausgabe = "1: absolut " + eins +
                  " Mal, relativ " + (eins/10) + " %\n";
    ausgabe = ausgabe + "2: absolut " + zwei +
                  " Mal, relativ " + (zwei/10) + " %\n";
    ausgabe = ausgabe + "3: absolut " + drei +
                  " Mal, relativ " + (drei/10) + " %\n";
    ausgabe = ausgabe + "4: absolut " + vier +
                  " Mal, relativ " + (vier/10) + " %\n";
    ausgabe = ausgabe + "5: absolut " + fuenf +
                 " Mal, relativ " + (fuenf/10) + " %\n";
    ausgabe = ausgabe + "6: absolut " + sechs +
                 " Mal, relativ " + (sechs/10) + " %";

    alert(ausgabe) ;
}// end of wuerfel
```

Aufgabe 4

```
function zweiWuerfel30Mal()  {
     var z1, z2;
     var ausgabe = "";
     for (var i = 1; i <= 30; i++)  {
        z1 = Math.ceil(6*Math.random());
        z2 = Math.ceil(6*Math.random());
        ausgabe = ausgabe + (z1 + z2) + "\n";
     }
     alert(ausgabe);
}
```

Aufgabe 5

```
function augenSumme()  {
   var zwei = 0;
   var drei = 0;
   var vier = 0;
   var fuenf = 0;
   var sechs = 0;
   var sieben = 0;
   var acht = 0;
   var neun = 0;
   var zehn = 0;
   var elf = 0;
   var zwoelf = 0;
   var z1, z2, summe;
   var ausgabe = "";

   for (var i = 1; i <= 1000; i++)  {
      z1 = Math.ceil(6*Math.random());
      z2 = Math.ceil(6*Math.random());
      summe = z1 + z2;
      switch(summe)   {
        case 2: zwei++; break;
        case 3: drei++; break;
        case 4: vier++; break;
        case 5: fuenf++; break;
        case 6: sechs++; break;
        case 7: sieben++; break;
        case 8: acht++; break;
        case 9: neun++; break;
        case 10: zehn++; break;
        case 11: elf++; break;
        case 12: zwoelf++;
      } // end of switch
   } // end of for

  ausgabe = ausgabe + "2: " + (zwei/10) + " %\n";
  ausgabe = ausgabe + "3: " + (drei/10) + " %\n";
  ausgabe = ausgabe + "4: " + (vier/10) + " %\n";
  ausgabe = ausgabe + "5: " + (fuenf/10) + " %\n";

  ausgabe = ausgabe + "6: " + (sechs/10) + " %\n";
  ausgabe = ausgabe + "7: " + (sieben/10) + " %\n";
  ausgabe = ausgabe + "8: " + (acht/10) + " %\n";
  ausgabe = ausgabe + "9: " + (neun/10) + " %\n";
```

```
    ausgabe = ausgabe + "10: " + (zehn/10) + " %\n";
    ausgabe = ausgabe + "11: " + (elf/10) + " %\n";
    ausgabe = ausgabe + "12: " + (zwoelf/10) + "%\n";

    alert(ausgabe) ;
  }// end of function
```

Aufgabe 6

```
function additionsaufgabe()  {
  var x = Math.round(20*Math.random());
  var y = Math.round(20*Math.random());
  var summe = x + y;
  var eingabe=parseInt(prompt("berechne "+x+"+"+y));
  if (eingabe == summe) alert("richtig!");
  else alert("falsch!");
}
```

Aufgabe 7

```
function subtraktionsaufgabe()  {
  var x = Math.round(20*Math.random());
  var y = Math.round(20*Math.random());
  var a = Math.max(x, y);
  var b = Math.min(x, y);
  var differenz = a - b;
  var eingabe =parseInt(prompt("berechne"+a+"-" +b));
  if (eingabe == differenz) alert("richtig!");
  else alert("falsch!");
}
```

Aufgabe 8

```
  function Notendurchschnitt()  {
    var note, summe = 0;
    for (var i=1; i<=3; i++)  {
      note = prompt("Gib die " + i + ". Note ein!", "");
      summe = summe + parseFloat(note);
    }
    durchschnitt = Math.round(10*summe/3) / 10;
    alert("deine Note ist: " + durchschnitt);
  }
```

Aufgabe 9

```
function rechne()  {
  var x, y, ergebnis, eingabe;
  var note = 1;
  for (var i=1; i<=5; i++)  {
    x = Math.floor(10*Math.random()+1);
    y = Math.floor(10*Math.random()+1);
    ergebnis = x*y;

    eingabe=parseInt(prompt("Wieviel ist"+x+"*"+y,""));
    if (eingabe != ergebnis)  note++;
  } // of for
  alert("deine Note ist: " + note);
}
```

Aufgabe 10

```
   function rechne()  {
     var x, y, ergebnis, operator, eingabe, zeichen;
     var note = 1;
     for (var i=1; i<=5; i++)  {
       x = Math.floor(10*Math.random()+1);
       y = Math.floor(10*Math.random()+1);
       operator = Math.floor(4*Math.random());
       switch(operator)   {
         case 0:
           zeichen = "+";
           ergebnis = x+y;
           break;
         case 1:
           zeichen = "-"; ergebnis = x-y; break;
         case 2:
           zeichen = "*"; ergebnis = x*y; break;
         case 3:
           zeichen = "/";
           ergebnis = x;
           x = x*y;
        } // end of switch
      eingabe = prompt("Wieviel ist " +x+ zeichen+y, "");
      if (eingabe != ergebnis)  note++;
    } // of for
    alert("deine Note ist: " + note);
  }
```

Aufgabe 11

```
function pythagoras()  {
  var a, b, c;
  a = parseInt(prompt("Gib 1. Dreiecksseite ein!", ""));
  b = parseInt(prompt("Gib 2. Dreiecksseite ein!", ""));
  c = parseInt(prompt("Gib 3. Dreiecksseite ein!", ""));
  if (a*a+b*b==c*c || a*a+c*c==b*b || b*b + c*c==a*a)
  alert("das Dreieck ist rechtwinklig");
  else alert("das Dreieck ist nicht rechtwinklig");
}
```

Aufgabe 12

```
   function Winkelumrechnung()  {
     var alpha, b;
     alpha = parseInt(prompt("Winkel in Grad?", ""));
     b = alpha * Math.PI / 180;
     alert("der Winkel im Bogenmaß beträgt: " + b);
   }
```

Aufgabe 13

```
<script>
  var alpha, b;
  b = parseInt(prompt("Winkel in Radiant?", "");
  alpha = b*180/Math.PI;
  alert("der Winkel in Grad beträgt: " + alpha);
</script>
```

Aufgabe 14

```
   function Dreiecksberechnung()  {
     var a, b, c, alpha, beta, gammaG, alphaG, betaG;
     a = parseFloat(prompt("Gib a ein!", ""));
     b = parseFloat(prompt("Gib b ein!", ""));
     c = parseFloat(prompt("Gib c ein!", ""));
     if (a+b <= c || a+c <= b || b+c <= a)
        alert("nicht möglich");
     else  {
       alpha = Math.acos((b*b+c*c-a*a)/(2*b*c));
       alphaG = 180*alpha / Math.PI;
```

```
      beta = Math.acos((a*a+c*c-b*b)/(2*a*c));
      betaG = 180*beta / Math.PI;
      gammaG = 180 - alphaG - betaG;
      alert("die Winkel im Gradmaß sind: \n" +
             alphaG + "\n" + betaG + "\n" + gammaG);
    }
  }
```

Aufgabe 15b

```
function Lotto()  {
  var a, b, c, d, e, f;
  a = Math.floor(49*Math.random())+1;
  do b = Math.floor(49*Math.random())+1;
    while (b == a);
  do c = Math.floor(49*Math.random())+1;
    while (c==b || c==a);
  do d = Math.floor(49*Math.random())+1;
    while (d==c || d==b || d==a);
  do e = Math.floor(49*Math.random())+1;
    while (e==d || e==c || e==b || e==a);
  do f = Math.floor(49*Math.random())+1;
    while (f==e || f==d || f==c || f==b || f==a);
  alert(a+"\n" + b+"\n"+c+"\n"+d+"\n"+e+"\n" + f);
}
```

Aufgabe 16

```
function geburtstag()  {
  var w = 0; // w=Wahrscheinlichkeit
  var gw = 1; // gw = Gegenwahrscheinlichkeit
  var ausgabe = "";
  for (var n=2; n<=30; n++)  {
    gw = 1;
    for (var i = 2; i<=n; i++) gw= gw * (366-i)/365;
    w = 1-gw;
    w = Math.round(100*w)/100;
    ausgabe= ausgabe + n + " Schueler: " + w + "\n";
  }  // End of for n
  alert(ausgabe);
}  // End of function
```

Untersuche, wie das folgende Programm funktioniert:

10 + Math.ceil(6*Math.random())	=	13

```
<html>
 <head>
  <script>
   function rechne()  {
     Rechner.ergebnis.value =eval(Rechner.anzeige.value);
   }
  </script>
 </head>

 <body>
  <form name = "Rechner">
    <input name="anzeige" type="text" size="50"
       style="font-size:20pt; font-weight: bold;
       background-color: silver;"  value ="">
         
    <input type = "text" size="3" value=" = " READONLY
       style="font-size:20pt; font-weight: bold; "
       onClick = "rechne()">
         
    <input name="ergebnis" type="text" size="20"
       style="font-size:20pt; font-weight: bold;
       background-color: silver;"  READONLY value ="">
  </form>
  </body>
</html>
```

Das Objekt *window*

In der Objekt-Hierarchie der Sprache *JavaScript* steht das Objekt *window* ganz oben, d.h. alle anderen Objekte (Tabellen, Bilder, Formulare usw.) sind untergeordnet. Das Objekt *window* ist das Basisfenster, welches im Browser erzeugt wird.

Das Objekt *window* kann weitere Fenster enthalten. Es kann geöffnet und geschlossen werden. Es besitzt eine Statuszeile, es kann mehrere Frames besitzen, es besitzt auf jeden Fall eine Chronik (*history*) mit Möglichkeiten, darin vor- und zurück zu navigieren. Es besitzt eine URL (*location*). Es besitzt eine Breite und eine Höhe und noch weitere Eigenschaften.

Das Objekt *window* besitzt eine Reihe von Methoden, zum Beispiel *alert(), setTimeout(), clearTimeout(), setInterval() usw*. Streng genommen müssten diese Methoden folgendermaßen aufgerufen werden:
```
window.alert("Hallo Welt");
```
Weil *Java-Script* jedoch nur eine sog. objektbasierte und keine objektorientierte Sprache ist, genügt auch die kurze Angabe `alert("Hallo Welt");`

Ein weiteres Unterobjekt von *window* ist der eigentliche Inhalt, das Objekt *document*.

Das Objekt *document*

Das *document*−Objekt bezieht sich auf den Inhalt des übergeordneten Objektes *window*, der in einem Browser-Fenster angezeigt wird.

Im folgenden Beispiel soll die Hintergrundfarbe geändert werden. Die Farbe wird durch eine 6-stellige Hexadezimalzahl eingestellt. Beachte jedoch die beiden unterschiedlichen Programmversionen!

```
<html>
  <head>
    <script>
      window.document.bgColor = "#FF0000";
    </script>
  </head>

  <body >
    Beachte die Hintergrundfarbe!
  </body>
</html>
```

In der obigen Version funktioniert die Farbeinstellung leider nicht. Das liegt daran, dass der Programmcode der geschriebenen Reihenfolge nach umgesetzt wird. Zuerst wird im obigen Beispiel der Java-Script-Teil, danach der HTML-Teil ausgeführt. Mit dem *body*-Tag wird jedoch die im Browser voreingestellte, weiße Hintergrundfarbe (und noch weitere Voreinstellungen) eingestellt.

Das nächste Beispiel liefert den gewünschten Farbeffekt:

```
<html>
  <head>
  </head>

  <body >
    <script>
      window.document.bgColor = "#FF0000";
    </script>
    Beachte die Hintergrundfarbe!
  </body>
</html>
```

Da man üblicherweise jedoch den *Java-Script*-Code im Head-Teil unterbringen möchte, bietet sich auch folgende Variante an:

```
<html>
  <head>
    <script>
      function farbeinstellung()  {
        window.document.bgColor = "#FF0000";
      }
    </script>
  </head>

  <body onLoad = "farbeinstellung()">
    Beachte die Hintergrundfarbe!
  </body>
</html>
```

Im Folgenden wird die Hintergrundfarbe im Sekundentakt geändert.

```
<script>

  function setColor()  {
    farbe = "#";
    for (var i=1; i<=6; i++) {
      n = Math.floor(16*Math.random());
      switch(n)   {
        case 0:
          ch = "0";
          break;
        case 1:
          ch = "1";
          break;
        ……
        case 10:
          ch = "A";
          break;
        case 11:
          ch = "B";
          break;
        ……
        case 15:
          ch = "F"
      }  // end of switch
      farbe = farbe + ch;
    }  // end of for
    window.document.bgColor = farbe;
    window.setTimeout("setColor()", 1000);
  }
</script>
```

Bemerkung: Analog lässt sich auch die Textfarbe einstellen.
Beispiel: `window.document.fgColor = "#00FF00";`

Mit **`document.getElementById()`** lässt sich auf ein HTML-Element zugreifen, welches ein *id-Attribut* besitzt.

```
<html>
  <head>
    <script>
      function ausrichten(wie) {
        document.getElementById("unentschlossen").align = wie;
      }
    </script>
  </head>

<body>
  <h1 id="unentschlossen">Wo gehöre ich eigentlich hin?</h1>
  <a href="javascript:ausrichten('left')">links?</a><br />
  <a href="javascript:ausrichten('center')">zentriert?</a><br />
  <a href="javascript:ausrichten('right')">rechts?</a><br />
</body>
</html>
```

Bemerkung: Manche HTML-Elemente besitzen ein Attribut „id“, andere besitzen das Attribut „name“. Für den letzteren Fall gibt es die analoge *Java-Script*-Methode **`getElementsByName()`**

Mit **`document.open()`** wird ein Dokument zum Schreiben geöffnet. Dabei wird kein Fenster geöffnet, sondern der (bisherige) Fensterinhalt zum Neubeschreiben freigegeben. **Eventuell vorhandener alter Inhalt geht dabei verloren**, lässt sich aber mit dem Back-Pfeil des Browsers wieder herstellen.

```
<html>
 <head>
   <script>
     function machNeu() {
       document.open();
       document.write("Hallo");
       document.close();
     }
   </script>

 <body>
   Dies ist ein Test <br />
   <a href="javascript:machNeu()">Dokument erneuern</a>
 </body>
</html>
```

Mit ***`document.close()`*** wird dem Browser signalisiert, dass der Aufbau der Seite abgeschlossen ist. Ansonsten würde sich der Browser so verhalten, als müsse er weiterhin noch die Seite laden.

Mit **`document.write()`** wird zusätzlicher Text **in den Html-Quelltext** des Dokumentes eingefügt und anschließend als solcher vom Browser interpretiert.. Der eingegebene Text wird als Parameter der *write*-Methode übergeben. Dabei muss ein Schrägstrich / in schließenden HTML-*Tags* bei der Ausgabe mit dem Zeichen \ maskiert werden. Die folgende Anweisung fügt das Zeilenumbruch-Tag in den Html-Quelltext ein: **`window.document.write("<br \/>")`**

Das nachfolgende Programm liefert etwa folgendes Ergebnis:

Hallo Susanne
dein Hobby ist Musik

```
<script>
  vorname = prompt("Bitte deinen Vornamen angeben:", "Willi");
  hobby = prompt("Bitte dein Hobby angeben:", "Mathematik");
  document.write("<b>Hallo " + vorname + "<\/b><br \/>" +
                                    "dein Hobby ist " + hobby);
</script>
```

Hinweis: Statt des Pluszeichens in der write-Anweisung kann man auch ein Komma benutzen.

Die Methode **document.writeln()** wirkt wie die *write*-Methode, fügt aber einen zusätzlichen Zeilenumbruch in den Html-Quelltext ein. Das ist nicht dasselbe wie das Einfügen eines **
-**Tags !
Normale Zeilenumbrüche innerhalb eines Html-Quellcodes werden vom Browser nur als Leerzeichen umgesetzt.
Betrachte dazu nachfolgendes Beispiel! Beachte dabei, dass Text, der zwischen den Tags **<pre>** und **</pre>** steht, vom Browser genauso übernommen wird wie er formatiert dargestellt ist.

```
<script>
  document.write("<pre>kein Umbruch ");
  document.writeln("erste Zeile");
  document.writeln("zweite Zeile <\/pre>");
  document.writeln("dritte Zeile");
  document.writeln("vierte Zeile");
 /script>
```

Obiger Text entspricht also folgendem Html-Code:

```
<pre>kein Umbruch erste Zeile
zweite Zeile </pre>
dritte Zeile
vierte Zeile
```

Der Browser liefert dazu folgende Ausgabe:

```
kein Umbruch erste Zeile
zweite Zeile
```

dritte Zeile vierte Zeile

Das nächste Programm ändert im Sekundentakt den Inhalt des ersten Html-Absatzes.

```
<head>
<script>
  var IntervalId = 0;

  function starte( )  {
    IntervalId = setInterval ( "fachAngabe()", 1000 );
  }

  function stoppe()  {
    clearInterval ( IntervalId );
  }

  function fachAngabe ( ) {
   if ( Math.random( ) > .5 )
    document.getElementById("Absatz1").innerHTML="Informatik";
   else
    document.getElementById("Absatz1").innerHTML = "Physik";
  }

</script>
</head>

<body>
  <div id="Absatz1" style="height: 1.5em; font-size: 2em;
    color: red;"></div>
  <input type="button" value="Start" onclick="starte()"/>
  <input type="button" value="Stop" onclick="stoppe()"/>
</body>
```

Das Objekt *history*

Im nächsten Programm wird der Besucher der Webseite nach dem dritten erfolglosen Versuch, das Passwort richtig einzugeben, auf diejenige Webseite zurückgeschickt, von der er hergekommen ist. Dazu dient die Anweisung "***history.back()***".
Die sog. *Escape*-Sequenz **"\n"** in der prompt-und alert-Anweisung bewirkt jeweils einen Zeilensprung (***n****ew line*). Die Escape-Sequenz **"\r"** würde das Einfügen einer Leerzeile (*carriage* ***r****eturn*) bewirken.

```
<script>
  function kontrolle()  {
    var eingabe="", i=0;
    while (eingabe != "Informatik")  {
      i=i+1;
      if (i<=3)  {
       // Hinweis: Strings müssen einzeilig sein!
        eingabe = prompt("Bitte Passwort
            eingeben...\nVersuch Nummer " +i,"");
        if (eingabe != null)  {
          if (eingabe != "Informatik")
            alert("falsch! \n Versuch Nummer: "+i);
          else  {
            alert("Passwort ist richtig");
            window.open("Seite2.htm");
            break;
          }
        }
      } // Ende von if i<=3
      else  {
        alert("Geh dahin zurück, wo du hergekommen bist!");
        history.back();
        break;
      }
    } //Ende der while-Schleife
  }
</script>
```

weitere *History*-Methoden:

window.history.go(-3) würde das drittletzte Dokument in der History-Liste laden.
window.history.back() ist identisch mit *window.history.go(-1)*
window.history.forward() ist identisch mit *window.history.go(1)*

Aufgabe 1: Suche (Google!) ein kleines Bild mit der Aufschrift *BACK* und platziere es unten auf deiner zweiten HTML-Seite. Wird dieses Bild angeklickt, so wird zur vorhergehenden Seite zurückgesprungen.

Fenster auf der Webseite

```
<script>
   function fenster()   {
     var win;
     win = window.open("", "NeuesFenster",
             "width = 300, height = 200, left = 200,
                    top = 300, resizable = yes");
     // Beachte: kein String darf über 2 Zeilen gehen
     win.document.open("text/plain");
     // win.document.open("text/html");
     win.document.write("Zeile 1 ");
     win.document.write("Fortsetzung Zeile 1<br \/>");
     win.document.write("Zeile 2");
     win.setTimeout('window.close()',5000);
    }
  </script>
```

Das neue Fenster wurde als normales Textfenster geöffnet.

Die Variable für das neue Fensterobjekt lautet *win*. Im *window.open* Befehl steht ein sog. Leerstring. Dies bedeutet, dass keine Datei in das Fenster geladen wird, sondern es soll ein neues, leeres Fenster werden. Wenn man hier stattdessen einen Dateinamen angibt, so wird die entsprechende Datei in das neue Fenster geladen. Das funktioniert leider (in 2018) nur mit dem Internet-Explorer und nur mit pdf-Dateien. Außerdem besitzen die weiteren Schreibanweisungen für das *win.document* dann keine Wirkung mehr.

Der Name des neuen Fensters ist *NeuesFenster*. Er muss aus einem einzigen Wort bestehen. Dieser Name wird benötigt, wenn man z.B. mit Hilfe eines Links, der nicht in diesem Fenster steht, eine Html-Seite in genau dieses neue Fenster laden will:

```
<a href="test.htm" target="NeuesFenster">Linktext</a>
```

Die Mindestbreite und -höhe für ein Fenster ist jeweils 100.

Das Aufrufen obiger Java-Script-Funktion lässt sich bekanntlich auch durch einen Link ausführen:

```
<a href="javascript: fenster()"> Fenster öffnen </a>
```

Natürlich kann man auch mehrere Fenster gleichzeitig öffnen. Dazu benötigt man nur mehrere Fenster-Variablen.

```
   var win1
   win1 = window.open("test.htm", "NeuesFenster",
     "width=310, height = 220, left = 200,top =300");
```

Im nächsten Programm wird gleich nach dem Starten der Seite ein Fenster im Sekundentakt hin und her springen. Beachte, daß *win* diesmal eine **globale Variable** ist!

```
</html>
  <head>
    <script>
      var win, a=1;
      win = window.open
        ("", "NeuesFenster", "width = 300,height = 200,
                              left = 400, top = 300");
      // Beachte, dass der String nur in einer Zeile stehen darf!
      win.document.write("Test-Fenster");

      var zeit = window.setInterval("spring()",1000)

      function spring()  {
        win.moveBy(100*a,50*a);
        a=-a;
      }

      function stoppen()  {
        window.clearInterval(zeit);
        win.close();
      }
   </script>
  </head>

  <body>
    <a href="javascript: stoppen()"> Fensterbewegung
        beenden </a>
  </body>
</html>
```

Das folgende Programmstück zeigt, dass sich eine Funktion auch ***rekursiv*** aufrufen lässt. Hier wird das Fenster namens ***win*** (Erzeugung siehe oben!) 20 mal um 1 Pixel nach rechts bewegt:

```
var i = 0;
function vor()  {
   if(i <= 20)  {
     i++;
     win.moveBy(1,0);
     setTimeout("vor()", 100);
   }
}
```

Statt des relativen Sprunges *moveBy(100, 50)* kann man auch einen absoluten Sprungbefehl verwenden: *moveTo(227, 310).* Hierbei entspricht die obere linke Fensterecke den angegeben Koordinaten.

Analog kann man auch die Größe des Fensters relativ oder absolut verändern: *win.resizeBy(-100, 80)* oder *win.resizeTo(350,250)*

Das folgende Beispiel lässt den Bildschirm etwas wackeln:

```
<script>
 function shake()   {
   for (var i = 10; i > 0; i--)  {
     for (var j = 50; j > 0; j--)  {
       window.moveBy(0,i);
       window.moveBy(i,0);
       window.moveBy(0,-i);
       window.moveBy(-i,0);
     }
   }
 }
</script>
```

Obiges Programm funktionierte 2018 leider nur im Internet Explorer.

Mit dem folgenden Programm lässt sich kurzzeitig ein neues Fenster öffnen.

```
<head>
 <script>
   var nF;
   function schliessen()  {
     nF.close();
   }

   function neuesFenster() {
     var nF = window.open("", "meinFenster", "width=200,
                                                 height=100");
     nF.document.write("<p>Dies ist das neue Fenster</p>");
     setTimeout("schliessen()", 3000);
   }
 </script>
</head>

<body>
  <p>Nach Klick auf den Button oeffnet sich fuer 3 Sekunden
     ein neues Fenster</p>
  <button onclick="neuesFenster()">Oeffne ein neues
                                              Fenster</button>
</body>
```

<u>Aufgaben</u>

Beachte bei den folgenden Aufgaben, dass das zu erzeugende neue Fenster eventuell hinter dem eigentlichen Browserfenster entsteht, so dass es gar nicht beobachtet werden kann. Mache deshalb das eigentliche Browserfenster so klein wie möglich, bevor du die zu programmierenden Fensterbewegungen startest!

1. Erzeuge ein Fenster, welches immer abwechselnd größer und kleiner wird! Benutze dafür den ***resizeBy***-Befehl.

2. Erzeuge ein Fenster, welches sich immer abwechselnd runter und hoch bewegt! Benutze dafür den ***moveTo***-Befehl.

3. Erzeuge in der Bildschirmmitte ein Fenster, welches immer größer wird! Benutze dafür die Befehle ***moveTo(…)*** und ***resizeBy(...)*** ! Das Fenster soll immer zentriert bleiben.

4. Ein Fenster soll sich „im Quadrat“ bewegen, d.h. es soll erst 100 Pixel nach rechts gehen, dann 100 Pixel nach unten, anschließend 100 Pixel nach links und zum Schluss 100 Pixel nach oben.

Lösung Aufgabe 1:

```
<script>
   var win, a=1;
   win = window.open
     ("", "NeuesFenster", "width=300, height=200,
                                   left=400, top=100");
   win.document.write("Test-Fenster");


   function groesser() {
      if (!win.closed) win.resizeBy(1,1);
      a++;
      if (a<100) window.setTimeout("groesser()",10);
      else kleiner();
   }


   function kleiner() {
      if (!win.closed) win.resizeBy(-1,-1);
      a--;
      if (a>0) window.setTimeout("kleiner()",10);
      else groesser();
   }


  </script>
 </head>

 <body>
   <a href="javascript: groesser();"> Fensterbewegung
   </a>
 </body>
</html>
```

Lösung Aufgabe 2:

```
<script>
   var win, a=1;
   win = window.open
     ("", "NeuesFenster", "width=300, height=200,
                                  left=400, top=100");
   win.document.write("Test-Fenster");


   function runter() {
      if (!win.closed)  win.moveTo(300, a)
      a++;
      if (a<100) window.setTimeout("runter()",10);
      else hoch();
   }


   function hoch() {
      if (!win.closed)  win.moveTo(300, a)
      a--;
      if (a>0) window.setTimeout("hoch()",10);
      else runter();
   }


  </script>
 </head>

 <body>
   <a href="javascript: runter();"> Fensterbewegung </a>
 </body>
</html>
```

Lösung Aufgabe 3:

```
<script>
   var win, a=1;
   win = window.open
     ("", "NeuesFenster", "width=300, height=200,
                                  left=300, top=150");
   win.document.write("Test-Fenster");


   function groesser() {
    if (!win.closed)  {
      win.moveTo(300-a, 150-a);
      win.resizeBy(2, 2);
    }
    a++;
    if (a<100) window.setTimeout("groesser()",10);
   }

  </script>
 </head>

 <body>
   <a href="javascript: groesser();"> Fensterbewegung
   </a>
 </body>
</html>
```

Lösung Aufgabe 4:

```
<script>
  var win, a=1;
  win = window.open("", "NeuesFenster", "width=300,
                                height=200, left=300, top=150");
  win.document.write("Test-Fenster");


  function nachRechts() {
    if (!win.closed) win.moveTo(300+a, 150);
    a++;
    if (a<100) window.setTimeout("nachRechts()",10);
    else {
      a =1;
      nachUnten();
    }
  }


  function nachUnten() {
    //Voraussetzung: beim ersten Mal ist a=1
    if (!win.closed) win.moveTo(400, 150+a);
    a++;
    if (a<100) window.setTimeout("nachUnten()",10);
    else nachLinks();
  }


  function nachLinks() {
    //Voraussetzung: beim ersten Mal ist a=100
    if (!win.closed) win.moveTo(300+a, 250);
    a--;
    if (a>0) window.setTimeout("nachLinks()",10);
    else nachOben();
  }


  function nachOben() {
    //Voraussetzung: beim ersten Mal ist a=0
    if (!win.closed) win.moveTo(300, 250-a);
    a++;
    if (a<100) window.setTimeout("nachOben()",10);
  }

  </script>
 </head>

 <body>
   <a href="javascript: nachRechts();"> Fensterbewegung </a>
 </body>
</html>
```

Das Objekt *location*

Das Objekt *location* enthält in seiner Eigenschaft *href* die Web-Adresse des aktuell geladenen Dokumentes. Mit *JavaScript* kann diese Adresse geändert werden:

```
……….
 <script>
   function umleiten()    {
     location.href = "Seite2.html"
   }
 </script>
</head>

<body onLoad="window.setTimeout('umleiten()',5000)">
  Dies ist ein Test. Nach 5 Sekunden werden Sie umgeleitet
</body>
```

Bemerkung: der *onLoad*-Event wird nicht aufgerufen, wenn man die *Back*-Taste des Browsers betätigt.

Das Objekt *location* lässt sich auch nutzen, um andere Dateitypen auf der HTML-Seite darzubieten. Der Eigenschaft *location.href* wird einfach der Name der Datei zugewiesen, die heruntergeladen werden soll. Bei einigen Dateien wie z.B. **.docx* oder **.pdf* wird der Benutzer gefragt, ob die Datei mit einem vorhandenen Programm geöffnet oder auf der Festplatte gespeichert werden soll. Bei bekannten Dateitypen wie z.B. **.jpg*, **.mp3* oder **.txt* öffnet der Browser diese Datei sofort selbst. Was dem Browser bekannt ist, hängt auch von den sog. *plugins* des Browsers ab.

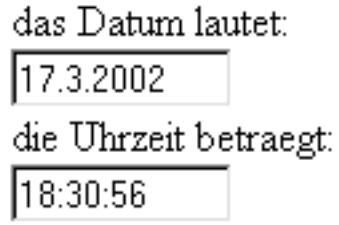
```
<script>
  function download()  {
    location.href="Information.docx"
  }
</script>
</head>

<body>
  <a href="javascript: download()"> Hier können Sie die
      Datei herunterladen </a>
</body>
```

Aufgabe: Der Internet-Surfer soll sich aus einem Menü eine von mehreren Dateien aussuchen können, die er herunterladen will. Schreibe das entsprechende Programm!

Lösung der Aufgabe:


```
<html>
 <head>
  <script>

   function download()   {
     var index, inhalt;
     index = document.Auswahl.Moeglichkeit.selectedIndex;
     inhalt =
      document.Auswahl.Moeglichkeit.options[index].value;
     location.href = inhalt;
   }

  </script>
 </head>

 <body>
  <form name = "Auswahl">
   <select name="Moeglichkeit" size="3" onChange="download()">
     <option selected value = "Seite1.doc"> Seite1
     <option value ="bild1.jpg"> Bild
     <option value = "BrownSugar.mp3"> Musik
   </select>
  </form>
 </body>
</html>
```

Datum und Uhrzeit auf der Homepage




```
<html>
 <head>
  <script>
    var jahr, monat, tag, stunden, minuten, sekunden;
    var aktuellesdatum;
    window.setInterval("uhr()", 1000);

    function uhr()  {
      aktuellesdatum = new Date();
      jahr = aktuellesdatum.getFullYear();
      monat = aktuellesdatum.getMonth()+1;
      tag = aktuellesdatum.getDate();
      stunden = aktuellesdatum.getHours();
      minuten = aktuellesdatum.getMinutes();
      sekunden = aktuellesdatum.getSeconds();
      //das Folgende bezieht sich auf das unten erstellte Formular
      window.document.Datumsformular.Datumsangabe.value =
         tag + "." + monat + "." + jahr;
      window.document.Datumsformular.Uhrzeit.value =
         stunden + ":" + minuten + ":" + sekunden;
    }
  </script>
 </head>
 <body>
   <form name = "Datumsformular">
     das Datum lautet:  <br />
     <input size = "10" name = "Datumsangabe"> <br />
     die Uhrzeit betraegt:  <br />
     <input size = "10" name = "Uhrzeit">
   </form>
 </body>
</html>
```

Das Schlüsselwort *new* erzeugt ein neues Objekt der Klasse *Date*. In diesem Falle wird also ein Objekt mit dem Namen *aktuellesdatum* erzeugt.Derartige Objekte besitzen mehrere Methoden, welche Informationen angeben (nur) über den Zeitpunkt, in dem sie erzeugt worden sind. Diese Informationen aktualisieren sich also nicht von selbst!

Aufgabe: Die Darstellung der einzelnen Zahlen im obigen Programm soll immer zweistellig sein!

Lösung

Möglichkeit 1:

// Nur die Funktion uhr() wird angepasst:

```
function uhr()  {
  aktuellesdatum = new Date();
  jahr = aktuellesdatum.getFullYear() - 2000;
  monat = (aktuellesdatum.getMonth()+1);
  if (monat < 10) monat = "0" + monat;
  tag = aktuellesdatum.getDate();
  if (tag < 10) tag = "0" + tag;
  stunden = aktuellesdatum.getHours();
  if (stunden < 10) stunden = "0" + stunden;
  minuten = aktuellesdatum.getMinutes();
  if (minuten < 10) minuten = "0" + minuten;
  sekunden = aktuellesdatum.getSeconds();
  if (sekunden < 10) sekunden = "0" + sekunden;
  // der Rest ist wie im obigen Programm.
}
```

Möglichkeit 2:

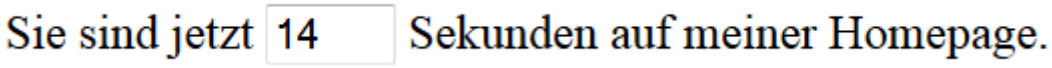
// Nur die Funktion uhr() wird angepasst:

```
function uhr()  {
  aktuellesdatum = new Date();
  jahr = "" + aktuellesdatum.getFullYear();
  jahr = jahr.substr(2,2);
  monat = "" + (aktuellesdatum.getMonth()+1);
  if (monat.length == 1) monat = "0" + monat;
  tag = "" + aktuellesdatum.getDate();
  if (tag.length == 1) tag = "0" + tag;
  stunden = "" + aktuellesdatum.getHours();
  if (stunden.length == 1) stunden = "0" + stunden;
  minuten = "" + aktuellesdatum.getMinutes();
  if (minuten.length == 1) minuten = "0" + minuten;
  sekunden = "" + aktuellesdatum.getSeconds();
  if (sekunden.length == 1) sekunden = "0" + sekunden;
  // der Rest ist wie im obigen Programm.
}
```

Aufgabe:

Vergleiche die beiden unterschiedlichen Berechnungen zur zweistelligen Jahreszahl! Beachte, dass wir erst vor relativ kurzer Zeit einen Jahrtausendwechsel hatten! Vielleicht wird jetzt verständlich, warum man teilweise besorgt war, dass evtl. wichtige Computerprogramme nicht mehr richtig funktionierten.

Wie lange war der Internet-Surfer auf der Web-Seite?



Im folgenden Programm werden zwei Objekte der Klasse *Date* benutzt, welche unter anderem die Anzahl der Millisekunden gespeichert haben, die seit dem 01.01.1970 vergangen sind. Diese Anzahl erhält man mit der Methode ***getTime()*** .

```
<html>
 <head>
  <script>
    var anfangszeit = new Date();
    var startzeit = anfangszeit.getTime();
    var differenz, aktuellesDatum;

    window.setInterval("Besuchszeit()", 1000);

    function Besuchszeit()   {
      aktuellesDatum = new Date();
      differenz= (aktuellesDatum.getTime() - startzeit)/1000;
      differenz= Math.floor(differenz);
      //das Folgende bezieht sich auf das Formular (s.u.)
      window.document.Zeitformular.Sekunden.value = differenz;
    }
  </script>
 </head>

 <body>
   <form name="Zeitformular">
     Sie sind jetzt
     <input size = "4" name="Sekunden">
     Sekunden auf meiner Homepage.
   </form>
 </body>
</html>
```

Man kann auch Zeitpunkte festlegen, indem man z.B. definiert:

```
var zeitpunkt = new Date (2019, 3, 9, 10, 30, 0)
```

Durch diese Festlegung entspricht die Variable *zeitpunkt* dem 09. **April** 2019 um $10.^{30}$ Uhr und 0 Sekunden.
Die Methode *zeitpunkt.getTime()* ermittelt dann wieder die Anzahl der Millisekunden, die seit dem 01.01.1970 bis zu diesem Termin vergangen sind.
Übrigens: bei der obigen Festlegung der Variablen *zeitpunkt* kann die Angabe der Uhrzeit auch fehlen.

Ähnlich wie im letzten Beispiel könnte man die Differenz zwischen der aktuellen Zeit und der Variablen *zeitpunkt* berechnen und das Programm entsprechend reagieren lassen. So könnten z.B. Teilprogramme erst nach Ablauf eines bestimmten Datums aufgerufen werden. Falls es nicht auf eine Differenz von Millisekunden ankommt, so betrachte nachfolgende vereinfachte Version. Diese macht sich zunutze, dass Objekte der Klasse *Date* auch größenmäßig miteinander verglichen werden können:

```
<html>
<head>
  <script>
    var stichtag = new Date(2019, 3, 10);
    var heute = new Date();

    function puenktlichkeit()   {
      if (heute < stichtag) alert("Willkommen!");
      else alert("zu spät!");
    }
  </script>
 </head>

 <body onLoad = "puenktlichkeit()">
 </body>
</html>
```

Aufgaben

1. Schreibe ein Programm, welches unter anderem alle 10 Sekunden kurz überprüft, ob ein bestimmter Zeitpunkt (z.B. heute 10.^{12} Uhr, orientiere dich dabei an der vom Computer angezeigten Zeit!) erreicht wurde. Falls ja, wird eine kurze Meldung ausgegeben und die weitere Überprüfung wird abgebrochen.
Hinweis: Benutze die Anweisungen ***x = window.setInterval(Funktion, ms)*** und ***window.clearInterval(x)*** .

2. Lege einen bestimmten Zeitpunkt fest (z.B. heute 8.^{42}Uhr)! In einem „Eingabefeld“ soll im Sekundentakt angegeben werden, wie viele Sekunden es noch bis zu diesem Stichzeitpunkt dauern wird (Countdown).

3. Die Reaktionszeit des Benutzers soll folgendermaßen gemessen werden: Nach dem Programmstart vergehen ein paar zufällige (zwischen 4 und 10) Sekunden. Dann erscheint (mit *alert*) eine Meldung, die mit der *Return*-Taste bestätigt werden muss. Der Zeitpunkt des Erscheinens der Meldung und auch der Zeitpunkt des Bestätigens werden festgehalten. Die Zeitdifferenz wird (in Sekunden, mit Nachkommastellen) ausgegeben.

4. Der Benutzer soll Multiplikationsaufgaben aus dem großen Einmaleins lösen, d.h. die beiden Faktoren liegen zwischen 10 und 20. Er soll drei Aufgaben lösen. Die durchschnittlich benötigte Zeit für eine Aufgabe wird ausgegeben (natürlich nur, wenn alle Aufgaben richtig gelöst wurden).

5. Schreibe eine *function warte(t)*, welche Folgendes bewirkt: Es wird t Millisekunden lang gewartet, bevor das Programm mit dem nächsten Befehl weiter macht.
Lösungsidee: Zu Beginn der Funktion wird in einer Variablen namens *startZeit* gespeichert, wie viele Millisekunden seit dem 01.01.1970 vergangen sind. Danach wird solange die ebenso definierte *endZeit* abgefragt, bis die Differenz *endZeit – startZeit* größer ist als der Parameter t in der Funktion *warte*. Damit wird diese Funktion beendet, sodass alle weiteren Befehle des Programms abgearbeitet werden können.
Teste dein Programm damit, dass erst nach etwa 5 Sekunden das Wort „Hallo“ (mit alert) ausgegeben werden soll!

Lösungen

Aufgabe 1

```
<html>
 <head>
  <script>
    // Die Variable stichtag sollte dem heutigen Datum entsprechen
    var stichtag = new Date(2019, 3, 14, 11, 32, 0);
    var x;  // Nur mit Hilfe der Variablen x bzw. durch die mit x zu
    realisierende Resetmöglichkeit lässt sich verhindern, dass die Funktion
    pruefung immer wieder aufgerufen wird.
    function start()  {
      x = window.setInterval("pruefung()", 10000);
    }

    function pruefung()   {
      var jetzt = new Date();
      if (jetzt > stichtag) {
        alert("Die Zeit ist um");
        window.clearInterval(x);
      }
    }

  </script>
 </head>

 <body onLoad = "start()">
 </body>
</html>
```

Aufgabe 3

```
<html>
 <head>
  <script>
    function reaktion()  {
      var x = 4000 + Math.floor(6000*Math.random());
      window.setTimeout("Messung()", x);
    }

    function Messung()   {
      var anfang = new Date();
      var startzeit = anfang.getTime();
      alert("RETURN drücken");
      var ende = new Date();
      var stopzeit = ende.getTime();
      var differenz = (stopzeit - startzeit)/1000;
      alert(differenz + " sek");
    }
  </script>
 </head>

 <body onLoad = "reaktion()">
   Bei Erscheinen der Eingabeaufforderung sofort die
   Returntaste drücken!
 </body>
</html>
```

Aufgabe 4

```
<html>
 <head>
  <script>
    var differenz, anzahlRichtig = 0, zeit = 0;

    function start()  {
      for (var i=1; i<=3; i++) aufgabe();
      zeit = Math.round(10*zeit/3)/10;
      if (anzahlRichtig == 3) alert(zeit + " sek");
      else alert("mindestens eine Lösung war falsch!")
    }

    function aufgabe()   {
      var x = Math.floor(9*Math.random()+11);
      var y = Math.floor(9*Math.random()+11);
      var ergebnis = x*y;

      var anfang = new Date();
      var startzeit = anfang.getTime();
      var eingabe = prompt(x + " * " + y +" = ", "");
      var ende = new Date();
      var stopzeit = ende.getTime();

      differenz = (stopzeit - startzeit)/1000;
      zeit = zeit + differenz;
      if (eingabe == ergebnis) anzahlRichtig++;
    }

  </script>
 </head>

 <body onLoad = "start()">
   Gib fuer die folgenden drei Aufgaben moeglichst
   schnell die richtigen Loesungen ein!
 </body>
</html>
```

Aufgabe 5

```
<html>
 <head>
  <script>
    function warte(t)  {
      var startDatum = new Date();
      var startZeit = startDatum.getTime();
      var endDatum = new Date();
      var endZeit = endDatum.getTime();
      var differenz = endZeit - startZeit;

      while (differenz < t) {
        endDatum = new Date();
        endZeit = endDatum.getTime();
        differenz = endZeit - startZeit;
      }

    }  // of function warte(t)

  </script>
 </head>


 <body onLoad = "warte(5000); alert('Hallo');">
 </body>
</html>
```

Bilder

```
<html>
 <head>
  <title>Grafik Wechsel</title>
  <script>
    function wechsel1()  {
      window.document.images[0].src = "Bild2.jpg";
    }

    function wechsel2()  {
      window.document.images[0].src = "Bild3.jpg";
    }

    function wechsel3()  {
      window.document.images[0].src =
                                    "originalBild.jpg";
    }
  </script>
 </head>

<body>
  <a href="http://www.spiegel.de"
    onMouseOver="wechsel1()" onMouseOut="wechsel2()"
    onClick="wechsel3()">
    <img src="originalBild.jpg">
  </a>
 </body>
</html>
```

Auf der obigen HTML-Seite gibt es nur ein einziges Bild. *Java-Script* nummeriert die Bilder auf einer Seite (beginnend bei 0). *images[0]* ist also das erste Bild auf der Seite.
Bei dem Wechsel der Bilder sollte man beachten, dass die Bilder gleich groß sind. Ansonsten kann es passieren, dass beim *MouseOver* des ersten Bildes zwar das zweite Bild erzeugt wird, welches aber sofort von der Maus wieder verlassen wird, weil es wesentlich kleiner als das erste Bild ist. Daraufhin wird sofort das dritte Bild gestartet.
Notfalls kann man folgende Anweisung benutzen:

```
<img src="originalbild.jpg" height="200" width="200">
```

// Dadurch werden alle Bilder automatisch angepasst (und verzerrt).
Beim Klick auf den Link (welcher ja ein Bild ist) wird natürlich zur angegebenen Webseite gesprungen, aber mit der *Back*-Taste des Browsers erscheint dann das in der Funktion *wechsel3* angegebene Bild wieder.

Die drei Funktionen *wechsel1*, *wechsel2* und *wechsel3* machen im Prinzip alle dasselbe. Das lässt sich dann wesentlich eleganter durch eine einzige Funktion mit Parametern lösen: *wechsel(a, bild)*
Der Parameter *a* enthält die Nummer des Bildes, der Parameter *bild* ist ein Objekt der Klasse *Image*. Im Folgenden ist nur der geänderte Quelltext abgedruckt:

```
  <script>
    var original = new Image();
    original.src ="originalBild.jpg";
    var Mausdrauf = new Image();
    Mausdrauf.src = "Bild2.jpg";
    var Mausweg = new Image();
    Mausweg.src = "Bild3.jpg";

    function wechsel(a, bild)   {
      window.document.images[a].src = bild.src;
    }
  </script>
 </head>

 <body>
  <a href="www.spiegel.de"
     onMouseOver="wechsel(0, Mausdrauf)"
     onMouseOut="wechsel (0, Mausweg)"
     onClick="wechsel(0, original)">
     <img src="originalBild.jpg">
  </a>
 </body>
</html>
```

Aufgabe: Ein Bild (Dateiname z.B.: *originalBild.gif*) soll schrittweise von rechts nach links springen.

Lösungsidee: Erzeuge mithilfe eines Bildbearbeitungsprogrammes (z.B. paint) ein leeres Bild mit dem Namen *leerBild.gif*, welches dieselbe Größe hat wie *originalBild.gif*.

Schreibe nun einen HTML-Code, welcher vier leere Bilder und das richtige Bild nebeneinander erzeugt. Schreibe anschließend eine *Java-Script* Funktion, welche sich nach jeweils einer Sekunde selbst aufruft und folgendes leistet: Das richtige Bild wird durch ein leeres Bild ersetzt und ein vorher leeres Bild (links daneben) wird durch das richtige Bild ersetzt. Dies erzeugt den Eindruck, als würde sich das Originalbild (nach links) bewegen.

```
<head>
<script>
  function bewegung()   {
    Bildsicherung = window.document.images[0].src;
    for (var a=0; a<=3; a++)
       window.document.images[a].src
                              = window.document.images[a+1].src;
    window.document.images[4].src = Bildsicherung;
    window.setTimeout("bewegung()", 1000);
  }
</script>
</head>

 <body onLoad="bewegung()">
    <img src ="leerBild.gif">
    <img src ="leerBild.gif">
    <img src ="leerBild.gif">
    <img src ="leerBild.gif">
    <img src ="originalBild.gif">
 </body>
```

Aufgaben

1. Verteile mehrere Bilder auf der Webseite. Diese Bilder sollen im Sekundentakt ihre Plätze tauschen!

2. Auf der Webseite steht ein relativ kleines Bild. Wenn man auf dieses Bild klickt, erscheint an derselben Stelle das gleiche Bild, nur wesentlich größer. Wenn man auf das große Bild klickt, erscheint wieder das kleinere Format. Hinweise: die Bilddatei muss zweimal vorhanden sein, eine Datei für das große Bild und eine für das kleine Bild. Die entsprechende Quelldatei wird ausgetauscht.
Es gibt leider nicht die Möglichkeit, die Quelldatei abzufragen, etwa `if (window.document.images[0].src == "BildGross.jpg")` Dies funktioniert nicht. Stattdessen kann man eine globale Variable als „Schalter“ benutzen. Beispiel: `var n=1`. In der „Bildwechselfunktion“ ändert man das Vorzeichen der Variablen durch den Befehl `n = -n`. Abhängig vom Vorzeichen von n kann man nun die Bildquelldatei festlegen.

3. Es soll eine kleine Dia-Show realisiert werden, die es dem Benutzer erlaubt, durch eine Reihe von Bildern vorwärts und rückwärts zu blättern. Es wird immer nur ein einziges Bild gezeigt. Erzeuge dafür zwei Buttons für das Vor- und Zurückblättern. Ein Button wird im *Body*-Teil zum Beispiel folgendermaßen erzeugt:

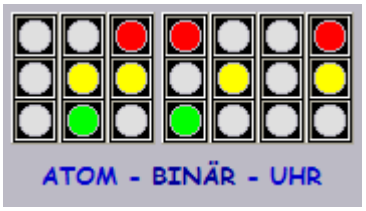
```
<input type="button" value="Vor" onClick="vorwaerts()">
```

4. Erweitere die Dia-Show nun derart, dass ein dritter Button zum Einsatz kommt, mit dem der Anwender ein aktuelles Bild vergrößern kann. Natürlich benötigt man dafür zusätzlich auch die entsprechend großen Bildquelldateien.

5. Countdown-Programm. Auf der Webseite erscheinen im Sekundentakt die Bilder der Ziffern 9, 8, …,1. Als letztes erscheint das (animierte) Bild eines Silvesterfeuerwerks.

6. Die Lösung dieser Aufgabe wird für geübte Programmierer sicherlich fast eine Stunde an Zeitaufwand erfordern. Aber auch gute Schüler mit euerem derzeitigen Kenntnisstand sollten diese Aufgabe lösen können (innerhalb mehrerer Stunden).

Auf der Homepage des Städtischen Gymnasiums in Kamen erscheint eine



digitale Uhr in der abgebildeten Form. Die Uhrzeit wird im sog. *BCD-Code* (**b**inary **c**oded **d**ecimal) dargestellt, d.h. jede Ziffer wird im Binärcode angezeigt.

Das Bild gibt die Uhrzeit 19 Uhr 35min 28sek an. Das Bild ändert sich natürlich jede Sekunde.

Lösungen

Aufgabe 2

```
<html>
 <head>
  <script>
   var n=1;

   function wechsel ()  {
    n = -n;
    if (n < 0) window.document.images[0].src ="BildGross.jpg";
    else window.document.images[0].src = "BildKlein.jpg";
   }

  </script>
 </head>

 <body>
  <img src="BildKlein.jpg"  onClick = "wechsel()">
 </body>
</html>
```

Aufgabe 3 *// nur vorwärts blättern*

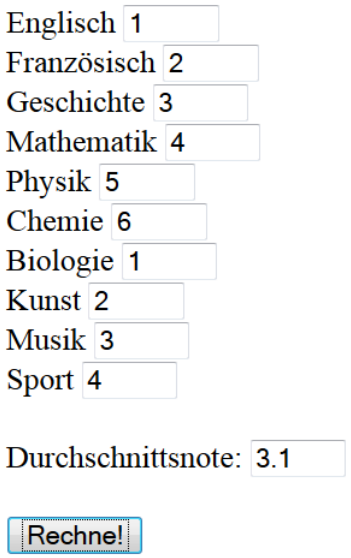
```
var a = 0;

function vorwaerts()  {
  a++;
  if (a>5) a=0;
  switch (a)   {
    case 0: window.document.images[0].src = "Auto0.jpg";break;
    case 1: window.document.images[0].src = "Auto1.jpg";break;
    case 2: window.document.images[0].src = "Auto2.jpg";break;
    case 3: window.document.images[0].src = "Auto3.jpg";break;
    case 4: window.document.images[0].src = "Auto4.jpg";break;
    case 5: window.document.images[0].src = "Auto5.jpg";
  }
}

<body>
  <img src="Auto0.jpg" height = "230">
  // Die Höhenangabe ist wichtig! Alle Bilder werden automatisch angepasst.
  <br />
  <input type="button" value="Vor" onClick="vorwaerts()">
  <input type="button" value="Zurueck" onClick="rueckwaerts()">
</body>
```

Aufgabe 5

```
<html>
 <head>
  <script>
    var a = 9;

    function countdown()  {
      Bilddateiname = "ziffer"+a+".jpeg";
      if (a>0)   {
        window.document.images[0].src = Bilddateiname;
        a--;
        window.setTimeout("countdown()", 1000);
      }
      else window.document.images[0].src = "feuerwerk.gif";
    }
  </script>
 </head>

 <body onLoad="countdown()">
   <p align = "center"> <img src="ziffer9.jpeg"> </p>
 </body>
</html>
```

Formulare auswerten

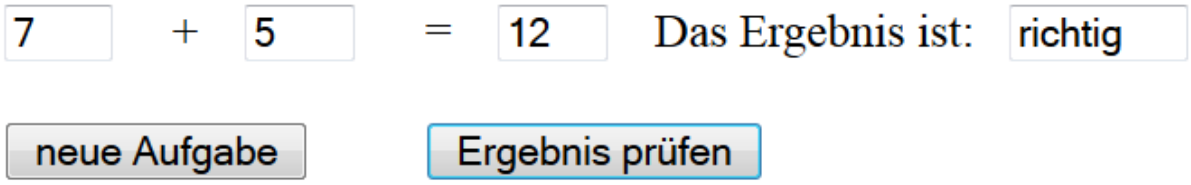
Im folgenden Beispiel geben wir Noten in ein Zeugnisformular ein und die Webseite soll die Durchschnittsnote berechnen. Da eine Webseite durchaus mehrere Formulare beinhalten kann, erhält jedes Formular im zugehörigen *form-Tag* (im HTML-Teil) einen eigenen Namen.




```
<html>
 <head>
  <script>
   function durchschnitt()  {
     var d = 0;
     for (var i=0; i<=9; i++)  {
       d=d+parseInt(window.document.Zeugnis.Fach[i].value);
     }
     d = d/window.document.Zeugnis.Fach.length;
     window.document.Zeugnis.Durchschnitt.value = d;
  }
</script>
</head>


<body>
 <form name="Zeugnis">
   Englisch <input type="text" name = "Fach" size="5"> <br />
   Französisch <input type="text" name="Fach" size="5"> <br />
   Geschichte <input type="text" name= "Fach" size="5"> <br />
   Mathematik <input type="text" name= "Fach" size="5"> <br />
   Physik <input type="text" name = "Fach" size="5"> <br />
   Chemie <input type="text" name = "Fach" size="5"> <br />
   Biologie <input type="text" name = "Fach" size="5"> <br />
   Kunst <input type="text" name = "Fach" size="5"> <br />
```

```
    Musik <input type="text" name = "Fach" size="5"> <br />
    Sport <input type="text" name="Fach" size="5"> <br /><br />
    Durchschnittsnote: <input type="text" size="5"
                                    name="Durchschnitt"> <br />
    <br />
   <input type="button" value="Rechne!" onClick="durchschnitt()">
  </form>

</body>
</html>
```

Wir wollen nun verschiedene Eingaben in Formularen auswerten. Jedes Eingabefeld innerhalb eines Formulars kann einen eigenen Namen erhalten. Im Folgenden sieht man ein kleines Übungsprogramm für Additionsaufgaben.



Betätigt man den Button *„neue Aufgabe“*, so wird eine neue Aufgabe mit Zufallszahlen gestellt. Das Ergebnis soll der Benutzer selbst eingeben. Man kann dieses Programm aber auch so nutzen, indem man einfach zwei Zahlen in die ersten beiden Felder eingibt (und das Ergebnis ins dritte Feld).

```
<script>
  function neueAufgabe()   {
    window.document.Rechner.s1.value = Math.floor(Math.random()*10);
    window.document.Rechner.s2.value = Math.floor(Math.random()*10);
    window.document.Rechner.Urteil.value = "";
    window.document.Rechner.sum.value = "";
  }

  function pruef()   {
    var zahl1 = parseInt(window.document.Rechner.s1.value);
    var zahl2 = parseInt(window.document.Rechner.s2.value);
    var ergebnis = zahl1 + zahl2;
    var eingabe= parseInt (window.document.Rechner.sum.value);
    var aussage = "";
    if (ergebnis == eingabe) aussage="richtig";
    else aussage = "falsch";
    window.document.Rechner.Urteil.value = aussage;
  }

</script>
</head>

<body>
 <form name = "Rechner">
   <input type = "text"  name="s1" size = "3"/>
       +   
   <input type = "text"  name="s2" size = "3"/>
        =   
   <input type = "text"  name="sum" size = "3" />   
   Das Ergebnis ist:   
   <input type = "text"  name="Urteil" size = "7" readonly/>
   <br /> <br />
   <input type = "button" value="neue Aufgabe"
                                        onClick="neueAufgabe()">
           
```

```
   <input type = "button" value="Ergebnis prüfen"
                                                onClick="pruef()">
 </form>
</body>
```

Aufgabe

Erstelle eine HTML-Seite, welche vier Formulare enthält: Im ersten kann man wie oben die Addition üben, im zweiten Formular die Subtraktion, im dritten die Multiplikation und im vierten die Division.

Achte bei der Division darauf, dass der Rechner nur Aufgaben wählt, deren Ergebnis ganzzahlig ist! Lösungsweg: Wähle zwei zufällige ganze Zahlen x und y, berechne deren Produkt $z = x \cdot y$ und stelle dem Besucher der Webseite die Aufgabe z : x

Achte bei der Subtraktion darauf, dass der Rechner nur Aufgaben wählt, deren Differenz nicht negativ ist! Tip: Mit Hilfe der Klasse *Math* kann man sehr einfach die größere bzw. kleinere zweier Zahlen bestimmen.

Im nächsten Beispiel werden ebenfalls verschiedene Eingaben in einem Formular ausgewertet.

```
<html>
<script>
 function pruefe()   {
  if (window.document.Anmeldung.Benutzer.value=="Administrator"
          && window.document.Anmeldung.PW.value=="Informatik")
    alert("OK");
  else  alert("Falsch!");
 }

 function loesche()   {
   window.document.Anmeldung.reset();
 }

 function initialisierung()   {
   window.document.Anmeldung.Benutzer.select();
   // window.document.Anmeldung.PW.focus()
 }
</script>
</head>

<body onLoad="initialisierung()">
 <form name = "Anmeldung">
  Benutzername:
  <input type="text" name="Benutzer" size="20" value="Willi">
  <br />
  Passwort: <input type = "password" name = "PW" size = "10">
  <br /> <br />
  <input type = "button" value="Pruefe, ob Admin sich
                 anmeldet" onClick = "pruefe()"><br /><br />
  <input type = "button" value="Alle Daten löschen"
                          onClick = "loesche()"><br /><br />
  <input type = "button" value="Benutzernamensfeld auswählen"
                                 onClick = "initialisierung()">
  <br /><br />
 </form>
 <br /> <br />
 </body>
</html>
```

Bemerkung: die Methode *focus* setzt den Fokus auf das entsprechende Eingabefeld. Die Methode *select* macht dasselbe, markiert aber zusätzlich den bisherigen Inhalt.

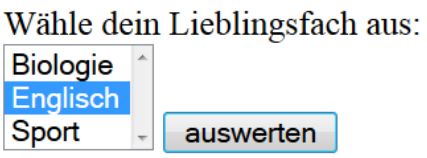
Das folgende Programm öffnet das E-Mail-Programm (falls vorhanden) des Anwenders, so dass die entsprechenden Daten verschickt werden können.

```
<html>
<head>
<script>
 function kontrolle()  {
   var zeichen = "0123456789-/ ", eingabe, laenge, ch;
   // beachte das Leerzeichen in zeichen!
   eingabe = window.document.forms[0].elements[2].value;
   laenge = eingabe.length;
   for (var n = 0; n < laenge; n++)   {
     ch = eingabe.charAt(n);
     if (zeichen.indexOf(ch) == -1)   {
       alert("keine gültige Telefonnummer!");
       window.document.forms[0].elements[2].value="";
       window.document.forms[0].elements[2].focus();
       break
     }
   } // end of for
 } // end of function
</script>
</head>

<body>
<form name = "Adresse"
  action = "mailto:test@xy.de?Subject= Gruesse vom Goethe"
    method = "post" enctype = "text/plain">
  <!-- wichtig: in der Angabe "mailto:test@xy.de?Subject=   darf kein einziges Leerzeichen
     stehen!  //-->
  Name:  <input type = "text" name="Familienname" SIZE = "20">
  <br /> <br />
  Wohnort: <input type="text" name="Ort" size="20">
  <br /> <br />
  Telefon: <input type="text" name="Telefon" size="20"
                                          onBlur = "kontrolle()">
   <!-- onBlur = wenn ein Element den Fokus verliert //-->
  <br /> <br />
  <input type = "submit" value = "abschicken!">
 </form>
</body>
</html>
```

Probleme mit deutschen Umlauten

Das E-Mail-Format ist *SMTP (simple mail transfer protocol)*, welches zunächst nur *ASCII*-Code verschicken konnte. Deshalb wurde die Erweiterung *MIME* geschaffen (**M**ultiple **I**nternet **M**ail **E**xtension). Wenn man nun z.B. Umlaute per E-Mail verschicken will, so muss man den richtigen *MIME*-Typ einstellen. Dies geschieht mit dem Befehl *enctype*.
Das Fragezeichen im obigen Beispiel ist wichtig!

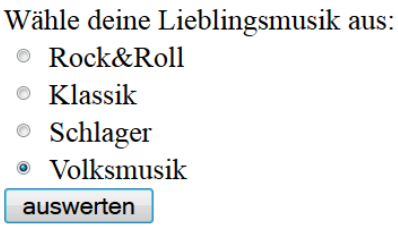
Auswahllisten auswerten



Das folgende Programm geht davon aus, dass nur ein einziger Fachname ausgewählt werden kann.

```
<script>
  var lieblingsfach = "0";

  function auswerten()   {
    var laenge = window.document.Faecherformular.fach.length;
    for (var i=0; i<laenge; i++)  {
      gewaehltesFach =
              window.document.Faecherformular.fach.options[i];
      if (gewaehltesFach.selected == true) {
         lieblingsfach = gewaehltesFach.value;
         break;
      }
    } // Ende von for

    var adjektiv = "";
    switch(lieblingsfach)   {
      case "0":
        adjektiv = "biologisch";
        break;
      case "1":
        adjektiv = "sprachlich";
        break;
      case "2":
        adjektiv = "sportlich";
        break;
      default: adjektiv = "anderweitig";
    } // end of switch

    alert("du bist " + adjektiv + " begabt");
   } // end of function auswerten()

</script>
</head>

<body>
 <form name = "Faecherformular">
   Waehle dein Lieblingsfach aus: <br />
   <select name = "fach" size = "3">
   <!-- Die Angabe size="1" besagt, dass nur ein Eintrag angezeigt wird. -->
     <option value = "0" selected> Biologie </option>
     <option value = "1"> Englisch </option>
     <option value = "2"> Sport </option>
   </select>

   <input type = "button" value="auswerten" onClick="auswerten()">
 </form>
</body>
```

Radiobuttons auswerten

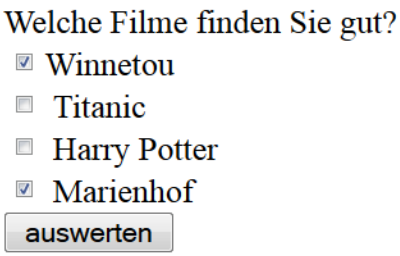



```
<head>
<script>

   function auswerten()    {
     var wert = "0";
     var auswahl;
     var aussage = "";
     var laenge = window.document.Musikformular.Art.length;
     for (var i=0; i<laenge; i++)   {
       auswahl = window.document.Musikformular.Art[i];
       if (auswahl.checked == true) {
         wert = auswahl.value;
         break;
       }
     } // Ende von for

     switch(wert)    {
      case "0":
        aussage = "Elvis lebt!";
        break;
      case "1":
        aussage = "Mozart lebt!";
        break;
      case "2":
        aussage = "Hallo again!";
        break;
      default: aussage = "Schön war die Zeit!";
     } // end of switch

     alert(aussage);
   }
</script>
</head>

<body>
 <form name = "Musikformular">
   Wähle deine Lieblingsmusik aus: <br />
   <!-- Wenn alle Radio-Buttons denselben Namen haben, dann kann nur einer von ihnen
   angeklickt werden. //-->
   <input type=radio name = "Art" value ="0"> Rock&Roll <br />
   <input type=radio name="Art" value="1" checked> Klassik<br />
   <input type=radio name = "Art" value = "2"> Schlager<br />
   <input type=radio name = "Art" value="3"> Volksmusik<br />
   <input type = "button" value="auswerten" onClick="auswerten()">
 </form>

</body>
</html>
```

Aufgabe

Programmiere einen Rechentrainer. Der Rechner stellt eine zufällige Aufgabe mit natürlichen Zahlen, der Benutzer gibt das Ergebnis ein, welches vom Rechner überprüft wird. Der Rechner gibt die Meldung „falsch“ oder „richtig“ aus. Die Aufgabenart (Addition, Subtraktion, Division, Multiplikation, Potenzrechnung, Zufallsrechenart) wird durch die Anwahl eines von 6 Radiobuttons bestimmt. Bei der Subtraktion ist darauf zu achten, dass das Ergebnis nicht negativ wird. Bei Divisionsaufgaben soll das Ergebnis ganzzahlig sein.

Checkboxen auswerten

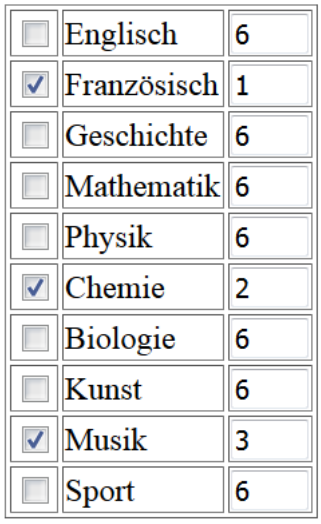



```
<html>
<head>
 <script>
   function auswerten()    {
     var aussage = "";

     if (window.document.Filmformular.Film[0].checked == true)
         aussage = aussage + "Karl-May-Filme sind super!\n";
     if (window.document.Filmformular.Film[1].checked == true)
         aussage = aussage + "Leonardo war toll\n";
     if (window.document.Filmformular.Film[2].checked == true)
         aussage = aussage + "Harry for president!\n";
     if (window.document.Filmformular.Film[3].checked == true)
         aussage = aussage + "Heimatfilme sind schön!";
     alert(aussage);

   }

</script>
</head>

<body>
 <form name = "Filmformular">
   Welche Filme finden Sie gut? <br />
   <input type = "checkbox" name = "Film"/> Winnetou <br />
   <input type ="checkbox" name="Film" checked/> Titanic<br />
   <input type ="checkbox" name = "Film"/> Harry Potter <br />
   <input type ="checkbox" name = "Film"/> Marienhof <br />
   <input type ="button" value="auswerten" onClick="auswerten()">
 </form>
</body>
</html>
```

Aufgabe

Oberstufenschüler können mehr Fächer belegen (die alle benotet werden) als eigentlich notwendig für das Bestehen des Abiturs ist. Die sog. Abiturdurchschnittsnote wird dann aus einer (möglichst optimalen) Auswahl dieser Fächernoten gebildet. Dabei gibt es natürlich Pflichtbedingungen, d.h. zum Beispiel: Mathematik und Deutsch sind auf jeden Fall dabei.

Schreibe nun ein Programm, welches diesen Sachverhalt simuliert! Die in der Abbildung erkennbaren Checkboxen bestimmen, ob das entsprechende Fach zur Durchschnittsnote beiträgt.



Hinweise zur Programmierung:
Das Obige ist offensichtlich eine Tabelle mit drei Spalten und 10 Zeilen.
Die Checkboxen haben alle den gemeinsamen Namen „Pflicht“, die Noten stehen alle in Input-Feldern mit dem gemeinsamen Namen „Note“.

Erzeuge zuerst eine Tabelle, die nur die Fächernamen enthält! Danach erweitere die Tabelle mit den Input-Feldern für die Noten! Danach programmiere die Berechnung der Durchschnittsnote aller Fächer! Anschließend erweitere die Tabelle um die Checkboxen! Dann sorge dafür, dass nur die angeklickten Fächer bei der Berechnung benutzt werden!

Lösung

```
<html>
 <head>
  <script>
   function durchschnitt()  {
     var d = 0;
     var anzahl = 0;
     for (var i=0; i<=9; i++)  {
       if (window.document.Zeugnis.Pflicht[i].checked == true)
        {
         anzahl++;
         d=d+parseInt(window.document.Zeugnis.Note[i].value);
        }
     }
     d = d/anzahl;
     window.document.Zeugnis.Durchschnitt.value = d;
   } // of function
  </script>
 </head>

 <body>
  <form name="Zeugnis">
    <table border>
      <tr>
        <td> <input type = "checkbox" name = "Pflicht"/> </td>
        <td> Englisch </td>
        <td> <input type="text" name = "Note" size="3"></td>
      </tr>
      <tr>
        <td> <input type = "checkbox" name = "Pflicht"/> </td>
        <td> Französisch </td>
        <td> <input type="text" name = "Note"  size="3"></td>
      </tr>
      <tr>
        <td> <input type = "checkbox" name = "Pflicht"/> </td>
        <td> Geschichte </td>
        <td> <input type="text" name = "Note"  size="3"></td>
      </tr>
      <tr>
        <td> <input type = "checkbox" name = "Pflicht"/> </td>
        <td> Mathematik </td>
        <td> <input type="text" name = "Note"  size="3"></td>
      </tr>
      <tr>
        <td><input type = "checkbox" name= "Pflicht"/> </td>
        <td> Physik </td>
        <td> <input type="text" name = "Note"  size="3"></td>
      </tr>
      <tr>
        <td> <input type = "checkbox" name = "Pflicht"/> </td>
        <td> Chemie </td>
```

```
      <td> <input type="text" name = "Note"  size="3"></td>
    </tr>
    <tr>
      <td> <input type = "checkbox" name = "Pflicht"/> </td>
      <td> Biologie </td> <td> <input type="text" name =
           "Note"  size="3"></td>
    </tr>
    <tr>
      <td> <input type = "checkbox" name = "Pflicht"/> </td>
      <td> Kunst </td>
      <td> <input type="text" name = "Note"  size="3"></td>
    </tr>
    <tr>
      <td> <input type = "checkbox" name = "Pflicht"/> </td>
      <td> Musik </td>
      <td> <input type="text" name = "Note"  size="3"></td>
    </tr>
    <tr>
      <td> <input type = "checkbox" name = "Pflicht"/> </td>
      <td> Sport </td>
      <td> <input type="text" name = "Note" size="3"></td>
    </tr>
  </table>
  <br /><br />


  Durchschnittsnote: <input type="text" size="3"
                                  name="Durchschnitt"> <br />
  <br />
  <input type="button" value="Rechne!"
                                    onClick="durchschnitt()">
 </form>

</body>
</html>
```

Frames

Im nächsten Beispiel wird eine Seite mit zwei Rahmen erstellt. Im unteren Rahmen (dieser wird die Seite *Navigation.htm* enthalten) kann man durch Anklicken von Radioknöpfen den Inhalt des oberen Rahmens (entweder die *Seite1.html* oder die *Seite2.html*) festlegen.

Zunächst der Quellcode derjenigen Seite, welche die Frameaufteilung festlegt:

```
<html>
 <head>
  <title> Frames </title>
 </head>

 <frameset rows = "90%, 10%">
  <frame name = "oben" src = "Seite1.html" noresize
    scrolling = "yes">
  <frame name = "unten" src = "Navigation.html" noresize>
 </frameset>
</html>
```

Nun die erste HTML-Seite, welche in den oberen Frame geladen werden soll:

```
<html>
 <head>
  <title>Seite1</title>
 </head>

 <body>
  Dies ist Seite 1  <!-- Analog gibt es eine Seite 2 -->
 </body>
</html>
```

Nun die Seite für den unteren Frame:

```
<html>
 <head>
  <title>Navigationsframe</title>
  <script>

   function seiteAufrufen(seite)   {
      parent.frames[0].location.href = seite;
   }
  </script>
 </head>

 <body>
  <form name = "Auswahl">
  <center>
    <!-- Die Radiobuttons sollen alle denselben Namen erhalten. Dann kann
         immer nur einer von ihnen gedrückt sein.  //-->
   <input type = "Radio" name = "RB"
        onClick = "seiteAufrufen('Seite1.html')"> Seite1
   <input type = "Radio" name = "RB"
        onClick = "seiteAufrufen('Seite2.html')"> Seite2
   <input type = "Radio" name = "RB"
      onClick="seiteAufrufen('http://www.spiegel.de')"> Seite3
  </center>
  </form>
 </body>
</html>
```

Alternativ zu den Radiobuttons kann man auch ein Pulldown-Menue zum Navigieren einsetzen. Dazu muss der Navigationsframe entsprechend geändert werden:

```
<head>
<title>Navigationsframe</title>
 <script>

  function seiteAufrufen()   {
    var index, inhalt;
    index=document.Auswahl.Moeglichkeit.selectedIndex;
    inhalt=document.Auswahl.Moeglichkeit.options[index].value;
    parent.frames[0].location.href=inhalt;
  }

 </script>
</head>

 <body>
  <form name = "Auswahl">
   <select name = "Moeglichkeit" size = "1">
    <option selected value = 'Seite1.html'> Seite1
    <option value = 'Seite2.html'> Seite2
    <option value = 'http://www.Spiegel.de'> Seite3
   </select>
   <input type=button value="Go!" onClick="seiteAufrufen()">
  </form>
 </body>
</html>
```

Bemerkung: Die Angabe **`size="1"`** besagt, dass nur ein Eintrag angezeigt wird.

Im obigen Beispiel kann man sich den *Go!-Button* und den Eventhandler *onClick* ersparen, wenn man statt dessen den Eventhandler *onChange* einsetzt. Dieser wird direkt im *select-Tag* eingesetzt:

```
<select name = "Moeglichkeit" size = "1"
   onChange = "seiteAufrufen()">
```

Arrays

```
<script>
 var  a=new Array(), b=new Array("Anton", "Fritz", 4711);

 function feld()  {
   for (var i=0; i<=2; i++)   {
    a[i] = i*i;
    alert(i+' '+a[i]+'  '+b[i])
   }
  }
</script>
</head>

<body onLoad="feld()">
.....
```

```
<script>
 var album=new Array("bild1.jpg", "bild2.jpg", "bild3.gif");
 var a=-1;

 function diashow()  {
   a++;
   window.document.images[0].src=album[a];
   if (a==2)  a=-1;
   window.setTimeout('diashow()',2000);
  }
</script>
</head>

<body onLoad="diashow()">
   <img src = "bild1.jpg">
.........
```

```
…….
<script>
 var pw = new Array(), eingabe, passwort;
 pw["vicki"] = "informatik";
 pw["michael"] = "goethe";
 pw["susanne"] = "gymnasium";

 function abfrage()  {
   var eingabe = prompt("Name?", "");
   eingabe = eingabe.toLowerCase();
   if ((eingabe=="vicki") || (eingabe == "michael")
        || (eingabe == "susanne"))
    {
     var passwort = prompt("Passwort?", "");
     passwort = passwort.toLowerCase();
     if (passwort == pw[eingabe])  {
       alert("OK");
       location.href = "start.html";
     }
     else alert("Falsches Passwort !");
    }
   else alert("Benutzer nicht vorhanden !");
  }
</script>
</head>
<body onLoad="abfrage()">
…..
```

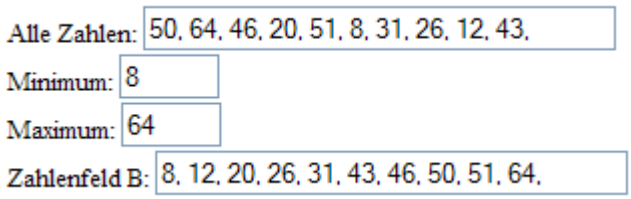
Aufgaben

1. Erstelle eine Dia-Show mit 10 Bildern!
a) die Dia-Show ist endlos lang. Sie wiederholt sich immer wieder.
b) die Dia-Show wird nach dem 10ten Bild beendet.

2. Erstelle eine Dia-Show, welche erst durch Anklicken eines Links aufgerufen wird!

3. Erstelle eine Dia-Show, die in einem Frame abläuft! Die Html-Seite ist unterteilt in einen linken und rechten Rahmen. Wenn man im linken Rahmen auf einen Link anklickt, läuft im rechten Rahmen eine Dia-Show ab.

4. Ein Array soll 10 zufällige, ganze Zahlen zwischen 0 und 100 enthalten.



a) Die einzelnen Zahlen des Arrays sollen in einem Textfeld eines Formulars durch Kommata getrennt ausgegeben werden.
b) Die kleinste und die größte Zahl des Arrays soll ausgegeben werden.
c) Die dritte und die fünfte Zahl des Arrays sollen vertauscht werden. Schreibe dafür eine Hilfsfunktion *function tausch(i, j)* , welche die i-te mit der j-ten Zahl im Array vertauscht. Gib zur Kontrolle das Array vor und nach dem Tausch aus. Beachte, dass die Feldnummerierung bei 0 anfängt.
d) Sorge durch eine oder mehrere Tauschaktionen dafür, dass die kleinste Zahl an der ersten Stelle im Array steht. Kontrolliere wieder!
e) Das gesamte Array soll der Größe nach sortiert werden.

5. Erstelle ein Array, welches 10 Namen enthält. Der Besucher der Homepage kann in einem Formular eine Zahl n zwischen 0 und 9 eingeben. Daraufhin wird ebenfalls in diesem Formular der n-te Name ausgegeben.

Lösungen

1.b) Die Lösung ist nahezu identisch mit dem Programm auf der Seite 109. Der wichtige Unterschied ist die if-Abfrage in der Funktion:

```
function diashow()  {
  a++;
  window.document.images[0].src=album[a];
  if (a<2)  window.setTimeout('diashow()',2000);
}
```

2. Die Lösung ist nahezu identisch mit dem Programm auf der Seite 109. Der Body-Teil lautet diesmal:

```
<body>
  <a href="javascript: diashow()"> DiaShow aufrufen </a>
  <img src = "bild1.jpg">
</body>
```

3. Die Lösung besteht aus drei HTML-Seiten:

```
<html>
 <frameset cols = "20%, 80%">
   <frame src = "SeiteLinks.htm">
   <frame src = "">
 </frameset>
</html>
```

die linke Seite (Dateiname: „SeiteLinks.htm“) lautet:

```
<html>
  <head>
    <script>
      function seite_aufrufen(seite)   {
        parent.frames[1].location.href = seite;
      }
    </script>
  </head>

 <body>
   <a href="javascript: seite_aufrufen('DiaShow.htm')">
      DiaShow aufrufen
   </a>
 </body>
</html>
```

Die rechte Seite (Dateiname: „DiaShow.htm“) lautet:

```
<html>
 <head>
  <script>
  var album=new Array("bild1.jpg", "bild2.jpg", "bild3.gif");
  var a=-1;

  function diashow()  {
    a++;
    window.document.images[0].src=album[a];
    if (a<2)  window.setTimeout("diashow()",2000);
  }
  </script>
</head>

<body onLoad = "diashow()">
  <img src = "bild1.jpg">
</body>
</html>
```

4.

```
<html>
<head>
<script>
  var  a = new Array();
  var  b = new Array();
  for (var i=0; i<10; i++) {
     zahl = Math.floor(Math.random()*100);
     a[i] = zahl;
     b[i] = zahl;
  }

  function teila()  {
    var ausgabe = "";
    for (var i=0; i<10; i++) ausgabe = ausgabe + a[i] +", ";
    Formular.Zahlenfeld.value = ausgabe;
  }

  function teilb()  {
    var min=a[0], max=a[0];
    for (var i=1; i<10; i++)  {
      if(a[i]<min)  min= a[i];
      if(a[i]>max)  max= a[i];
    }
    Formular.Minimum.value = "" + min;
    Formular.Maximum.value = "" + max;
  }

  function tausch(i,j) {
    var hilf = b[i];
    b[i] = b[j];
    b[j] = hilf;
  }

  function feldausgabeb()  {
    var ausgabe = "";
    for (var i=0; i<10; i++) ausgabe = ausgabe + b[i] +", ";
    Formular.Zahlenfeldb.value = ausgabe;
  }

  function teilc()   {
    tausch(3,5);
    feldausgabeb();
  }


  function teild()   {
    for (var j=1; j<10; j++)
       if (b[j] < b[0])  tausch(0,j);
    feldausgabeb();
  }
```

```
  function teile()   {
    for (var i=0; i<9; i++)
       for (var j=i+1; j<10; j++)
          if (b[j] < b[i])  tausch(i,j);
    feldausgabeb();
  }

</script>
</head>

<body onLoad="teila(); teilb(); teile();">
  <form name = "Formular">
    Alle Zahlen:
    <input type="text" name = "Zahlenfeld" size="35">  <br />
    Minimum:
    <input type = "text" name = "Minimum" size = "4">  <br />
    Maximum:
    <input type = "text" name = "Maximum" size = "4">  <br />
    Zahlenfeld B:
    <input type="text" name= "Zahlenfeldb" size="35">
  </form>
</body>
</html>
```

5.

```
<html>
<head>
<script>
   var  a=new Array("Anna", "Anton", "Berta", "Bob",
     "Corinna", "Dan", "Ella", "Erich", "Gerda", "Hans");
   var ausgabe = "";
   for (var i=0; i<10; i++) ausgabe = ausgabe + a[i] + ", ";

   function namensausgabe()  {
     Formular.Namensfeld.value = ausgabe;
     var n = Formular.Nummer.value;
     Formular.Vorname.value = a[n];
   }
</script>
</head>

<body>
  <form name = "Formular">
    Namen: <input type="text" name="Namensfeld" size = "60">
    <br />
    <input type = "text" name = "Nummer" size = "4"
            value = "0" onChange = "namensausgabe()">ter Name:
    <input type = "text" name = "Vorname" size = "20">  <br />
  </form>
</body>
</html>
```

Cascading Style Sheets (CSS)

```
<html>
 <head>
  <script>

    function wechsel(farbe,id,groesse)  {
      document.getElementById(id).style.color = farbe;
      document.getElementById(id).style.fontSize=groesse;
    }
  </script>
 </head>

 <body>
 <a id = "linkNummer1" href = "seite1.htm"
    //diese Seite 1 muss natürlich existieren.
    onmouseover = "wechsel('red', 'linkNummer1', '20pt')"
    onmouseout  = "wechsel('blue','linkNummer1','15pt')">
    //beachte die unterschiedlichen Hochkommata!
    Hier geht es zur Seite 1
 </a>
</body>
</html>
```

Im folgenden Programm wird ein Link als solcher erst ersichtlich, wenn man mit der Maus über ihn fährt. Ansonsten erscheint er wie ein ganz normaler Text.

```
<html>
<head>
  <style type = "text/css">
   .normal   <!-- hiermit wird eine Type-Klasse definiert. Beachte den Punkt! -->
    {
     font-Size: 12pt;
     color: black;
     text-decoration: none;
    }
  </style>

  <script>
    function wechsel(id, status)   {
      document.getElementById(id).style.textDecoration=status;
    }
  </script>
</head>

<body>
  Nur der folgende Name dient als Link. Unsere Schule ist nach
  <a class="normal" id="linkNummer1" href="Seite1.htm"
      onmouseover="wechsel('linkNummer1', 'underline')"
      onmouseout="wechsel('linkNummer1', 'none')">Goethe</a>
  benannt.
</body>
</html>
```

Das folgende Programm ändert kontinuierlich die Größe eines Absatzes.

```
<html>
<head>
<script>
  var a=10, gr, absatz;

  function start()   {
    absatz = document.getElementById("heiss");
    groesser();
  }


  function groesser()   {
    a++;
    if (a <= 25)  {
      gr = a + "pt";
      absatz.style.fontSize = gr;
      window.setTimeout("groesser()", 20);
    }
    else  kleiner();
  }


  function kleiner()  {
    a--;
    if (a >= 10)  {
      gr = a+"pt";
      absatz.style.fontSize = gr;
      window.setTimeout("kleiner()", 20);
    }
    else groesser();
  }


</script>
</head>

<body onload="start()">
 <p id = "heiss">Heisse Links:</p>
 <br>
 <a href = "seite1.html" >Hier geht es zur Seite 1</a>
</body>
</html>
```

Abschalten der rechten Maustaste

Mit dem folgenden Programm lässt sich die rechte Maustaste ausschalten. Damit ließe sich z.B. verhindern, dass von dieser Seite Bilder gespeichert werden. Das funktioniert so allerdings nur in Firefox und im MicroSoft Internet Explorer. Beachte, dass für andere Browser dieser Effekt natürlich nicht eintritt. Bedenke allerdings auch, dass diese Webseite (zusammen mit allen enthaltenen Bildern) im Cache des Surfers gespeichert sein muss. Dort kann man sehr leicht an die Bilder herankommen. Außerdem erhält man die Bilder auch mit Hilfe eines Screenshots. Noch einfacher wäre es, wenn der Surfer sein JavaScript abschaltet.

```
<html>
 <head>
  <script>
    if (navigator.appName == "Netscape")    {
      window.document.captureEvents(Event.mouseDown);
    }
    window.document.onmousedown = taste;


    function taste(evt)    {
      if (navigator.appName == "Netscape")
         if (evt.which == 3)  {
           alert("Na Na");
           return false;
         }
    }

  </script>
 </head>

 <body>
    Drueck doch mal auf die rechte Maustaste!
 </body>
</html>
```

Cookies

Ein Cookie schreibt sieben Einträge in eine Zeile. Einige dieser Einträge werden automatisch gesetzt. Die Reihenfolge der Einträge ist bei Mozilla und IE unterschiedlich. Dies spielt aber bei der Programmierung keine Rolle. Cookies werden auf dem Rechner des Surfers gespeichert. Der konkrete Speicherort hängt vom benutzten Browser und vom Betriebssystem ab.

Die sieben Einträge der Cookies sind:

- Domainname (desjenigen, der das Cookie geschrieben hat).
  TRUE/FALSE Verfügbarkeit des Cookies über die gesamte Domain.
- Verzeichnis, in dem der Programmierer dieses Cookie geschrieben hat. Nur wenn seine Homepage in diesem Verzeichnis bleibt, kann er mit diesem Cookie arbeiten.
- TRUE/FALSE Übertragung via SHTTP (**S**ecure **H**yper**T**ext **T**ransfer **P**rotocol = eine verschlüsselte Datenübertragung) möglich?
- Verfallsdatum des Cookies (codiert in Millisekunden)
- Name des Cookies
- Wert des Cookies

Nur die letzten drei Einträge sind eigentlich für den Programmierer interessant. Als Wert des Cookies lassen sich z.B. Zahlen oder längere Texte oder auch Datumsangaben einsetzen. Wichtig ist natürlich nur, dass der Programmierer die Bedeutung dieses Wertes kennt.

Beispiel:
www.aldi.de FALSE /Kunden FALSE 1041289200 testcookie 123

Das folgende Programm erzeugt ein Cookie, welches beim Surfer (falls vom Browser zugelassen) gespeichert wird:

```
<html>
<head>
<script>
  if (navigator.cookieEnabled == true)  {
     alert("Cookies sind erlaubt");
     CookieSetzen('Zaehler', '1', new Date(2018,1,28))"
  }
  else alert("Cookies sind nicht erlaubt");

  function CookieSetzen(name, wert, verfallsdatum)  {
    window.document.cookie = name + "=" + wert +
       "; expires = " + verfallsdatum.toGMTString();
  }
</script>
</head>

<body>
  Dies ist ein Cookie-Test
</body>
</html>
```

Im obigen Programm wird mit der Eigenschaft *cookie* des *document*-Objektes gearbeitet. Diese Eigenschaft besteht im Prinzip aus einem einzigen String, der alle aktuellen Cookies des Programmierers enthält (der Programmierer kann mehrere Cookies setzen). Durch die obige Funktion *CookieSetzen* wird diesem String ein weiteres Cookie angehängt.

Die Methode *toGMTString()* wandelt das Datum in einen String um.

Das erstellte Cookie wird beim Surfer nicht unter dem im Programm angegebenen Namen gespeichert. Der Internet Explorer benutzt als Dateinamen für ein Cookie den Verzeichnisnamen, in dem das Cookie (auf der Homepage-seite des Web-Servers) erstellt wurde. Wurden mehrere Cookies vom selben Verzeichnis aus erstellt, so werden sie alle aneinander gehängt und unter demselben Namen abgespeichert. Mozilla sammelt alle eintreffenden Cookies in einer einzigen Datei namens *cookies.txt*.

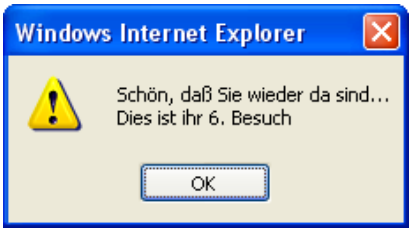
Die Browser überwachen selbsttätig alle gespeicherten Cookies und löschen diese, wenn deren Verfallsdatum abgelaufen ist. Will der Homepage-Programmierer ein Cookie beim Surfer löschen, so kann er dies nur machen, indem er dasselbe Cookie noch einmal schreibt, diesmal aber mit einem veralteten Verfallsdatum (am besten 1.1.1970).

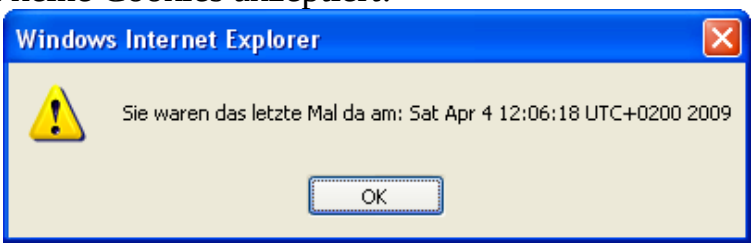
Zwei Cookies setzen

```
<html>
<head>
<script>
  var Datum = new Date();
  var verfall = new Date(2018,4,20);
  CookieSetzen("Counter", "1", verfall);
  CookieSetzen("Besuch", Datum, verfall);

  function CookieSetzen(name, wert, verfallsdatum)  {
    window.document.cookie = name + "=" +
     wert + ";expires =" + verfallsdatum.toGMTString();
  }
</script>
</head>
<body>
  Ich setze hier zwei Cookies
</body>
</html>
```

Auswertung mehrerer Cookies

Erinnerung: beachte bei dem Testen des folgenden Programmes, dass der Browser des Surfers eventuell keine Cookies akzeptiert!






```
<html>
<head>
<script>
var a=0;
var verfall = new Date(201,4,20);
var Cookies = document.cookie.split(";");
var CookiesAnzahl = Cookies.length;

var set = CookieAuslesen("Counter");
//Counter muss das erste abgespeicherte Cookie sein, sonst Leerzeichen wie bei " Besuch"
var besuchsDatum = CookieAuslesen(" Besuch");

if (set == false)   {
   alert("Willkommen bei dem ersten Besuch meiner Site!");
   document.cookie = "Counter=" + 1 + "; expires = "
                        + verfall.toGMTString();
   document.cookie = "Besuch=" +escape(new Date())+
                "; expires=" + verfall.toGMTString();
}
else   {
   set = parseInt(set) + 1;
   alert ("Schön, dass Sie wieder da sind...\nDies ist
             ihr " + set + ". Besuch");
   // die Strings dürfen eine Zeile nicht überschreiten
   document.cookie = "Counter=" + set + "; expires =
                            " + verfall.toGMTString();
   alert("Sie waren das letzte Mal da am: " + besuchsDatum);
   document.cookie = "Besuch=" +escape(new Date())+
                "; expires=" + verfall.toGMTString();
}

  function CookieAuslesen(name)  {
    for (a = 0; a < CookiesAnzahl; a++)   {
      aktuellerCookie = Cookies[a].split("=");
      if (aktuellerCookie[0] == name)   {
        return unescape(aktuellerCookie[1]);
        break;
      }
    }
    return false;
  }
```

```
</script>
</head>

<body>
  Zwei Cookies auslesen und neu schreiben
</body>
</html>
```

Aufgabe

1. In der obigen Datums- bzw. Zeitausgabe wird auch noch ein überflüssiger Wert für die sog. UTC (= **U**niversal **C**oordinated **T**ime = Weltzeit) angegeben. Entferne diese Angabe aus der entsprechenden Ausgabe, indem du den zugehörigen String am Ende abschneidest!

Objekte selbst erzeugen

Im Folgenden wird ein nicht vordefiniertes Objekt erzeugt (in anderen Programmiersprachen wird dies auch „Klasse“ genannt).

```
<html>
<head>
<script>

 function lehrer(name, fach1, fach2)  {
   this.name = name;
   this.fach1 = fach1;
   this.fach2 = fach2;
   this.zeige = show;
   function show()   {
     var win = window.open("", "Benutzerdaten");
     win.document.writeln("Nachname: "
                          + this.name + "<br \/>");
     win.document.writeln("Fach1: " + this.fach1);
     win.alert("weiter nach RETURN");
     win.close();
   }
 }

 var Meier = new lehrer("Meier", "Mathe", "Latein");
 var Horn = new lehrer("Horn", "Musik", "Kunst");

 alert(Meier.fach2);
 Meier.zeige();
</script>
</head>

<body>
  Gerade wurden auf zwei verschiedene Arten die
  <br />beiden Fächer von Herrn Meier dargestellt.
</body>
</html>
```

HTML-Seite ausdrucken

Der folgende Link bewirkt, dass die HTML-Seite ausgedruckt werden kann:

```
<a href="javascript:self.print()">Seite drucken</a>
```