Kleine Einführung in Javascript

1. Ein erstes Beispiel

JavaScript ist eine "einfache" Programmiersprache für das Internet, mit der man HTML-Seiten aufwerten kann. *Wo steht ein JavaScript in deinem HTML-Text?*
Einfache Antwort: entweder zwischen den beiden head-Tags, also im Kopf des Dokumentes, oder zwischen den body-Tags.

Beispiel 1: das sog. "Hallo-Welt-Programm":

```
<html>
<head>
<title>Javascript Programmbeispiel 1</title>
</head>
<body bgcolor=#00FFFF>
<h3>Testumgebung für einfache Javascript-Programme</h3>
<script language="JavaScript">
<!--
document.write("Hallo Welt-1!");
alert("Hallo Welt-2!");
//-->
</script>
</body>
</html>
```

Das Programm demonstriert zwei Möglichkeiten, Informationen auf dem Bildschirm auszugeben.
document.write("Hallo Welt-1!"); schreibt direkt in das HTML-Dokument.
alert("Hallo Welt-2!"); öffnet ein schickes Ausgabefenster und schreibt darin die Nachricht.

Javascript-Programme laufen komplett innerhalb des Browsers ab. Sie können nicht auf die Festplatte des Anwenders zugreifen. Daher sind Javascript-Programme sehr sicher. Sie können keine Programme oder Daten des Anwenders zerstören. Das schlimmste, was ein Javascript-Programm tun kann, ist eine Fehlermeldung auszugeben.

Javascript ist keine Teilmenge der Sprache Java. Obwohl Javascript ansatzweise ähnlich strukturiert ist wie Java, gibt es viele Unterschiede. Insbesondere ist Java ein **Compiler**, d.h. die Programme werden komplett in eine maschinennahe Sprache übersetzt. Im Gegensatz dazu werden Javascript-Programme von einem sog. **Interpreter** ausgeführt. Jede Programmzeile wird erst beim Aufruf vom Browserprogramm übersetzt und ausgeführt. Eine Folge davon ist, dass man den Quelltext von Javascript-Programme kaum vor den Blicken neugieriger Anwender (Schüler) verstecken kann.

2. Variablen

Variablen werden mit dem Schlüsselwort **var** deklariert. Variablen können **Zahlen** (z.B. 1.2 oder 3.45E6, **Texte** (strings, z.B. "GSG-Lebach") oder **bool'sche Werte** (true, false) annehmen. Im Gegensatz zu anderen Programmiersprache unterscheidet Javascript zwischen Groß- und Kleinschreibung. Zahl ist daher eine andere Variable als zahl!

Einfache lineare Programme enthalten gewöhnlich eine oder mehrerer **E**ingaben, einen **V**erarbeitungsteil und eine **A**usgabe. Es sind die typischen EVA-Programme. Das nächste Beispiel zeigt, wie mit Javascript zu einem einzugebenden Nettopreis die Mehrwertsteuer berechnet werden kann.

Beispiel 2: Berechnung der Mehrwertsteuer

```
<html>
<head>
<title>Javascript</title>
</head>
<body bgcolor="#CCFFFF" >
<blockquote>
<h1>Berechnung der Mehrwertsteuer</h1>
<script language="JavaScript">
<!--
//einfaches EVA-Programm
var Nettobetrag, MWSt
Nettobetrag=prompt(" Nettobetrag : ",2000);
MWSt = Nettobetrag*0.16;
document.write("<b>Nettobetrag :  ", Nettobetrag,"</b>","<br>");
document.write("Mehrwertsteuer :  ", MWSt,"<br>");
//-->
</script>
</blockquote>
</body>
</html>
```

Zur einfachen Eingabe von Werten in einer schicken Box dient die Anweisung prompt(...). Variable können auch direkt im Programm einen Wert erhalten.
Beispiel: var Nettobetrag = 2000;

Der sog. Zuweisungsoperator ist ein Gleichheitszeichen =.
Die arithmetischen Operatoren sind:

+	Addition
-	Subtraktion
*	Multiplikation
/	Division
%	modulo (Rest bei Ganzzahldivision)

Interessant ist die Möglichkeit, die Ausgabe mit den üblichen HTML-Tags zu formatieren. Vergleiche z.B. im obigen Beispiel die Formatierung als fett. Die Tags müssen dazu mit " " eingefasst sein.

3. Buttons

Typisch für Javascript ist die Möglichkeit, auf Ereignisse, wie beispielsweise das Anklicken eines Buttons zu reagieren. In der folgenden *"Spielerei"* werden drei Schaltknöpfe definiert, die jeweils eine andere Hintergrundfarbe festlegen.
Zunächst werden drei einfache eigene Funktionen mit den Namen red(), yellow() und blue() definiert. { ist der Beginn, } das Ende der Funktionsdefinition. Funktionen werden später noch ausführlicher behandelt.

Die drei Schaltknöpfe werden im body-Teil innerhalb des form-Tags (für Formular) definiert.
Beispiel: *<input type="button" value="rot" onClick="red()"/>*

Wird der mit dem Text "rot" beschriftete Knopf mit der Maus angeklickt, so wechselt die Hintergrundfarbe des Dokumentes! Natürlich kann man per Mausklick auch sinnvollere Funktionen aufrufen!

Beispiel 3: Farbwechsel per Knopfdruck

```
<html>
<title>Farben wechseln</title>
<head>
<script language="JavaScript">
<!--
function red() {
document.bgColor="red";
}
function yellow() {
document.bgColor="yellow";
}
function blue() {
document.bgColor="55ffff";
}
//-->
</script>
</head>
<body>
<h1>Farbwechsler</h1>
<form>
<input type="button" value="rot" onClick="red()" >
<input type="button" value="gelb" onClick="yellow()" >
<input type="button" value="blau" onClick="blue()" >
</form>
</body>
</htm>
```

4. Formulare

Die einfachen EVA-Programme wirken schöner, wenn man ein Berechnungsformular auf dem Bildschirm hat. In ein Eingabefeld gibt man immer wieder einen Wert ein und durch einen Mausklick auf einen Knopf erscheint im gleichen Formular in einer weiteren Zeile jeweils das Ergebnis. Die Berechnung der Mehrwertsteuer sieht dann schon fast professionell aus.

Beispiel 4: Mehrwertsteuer mit Formular

```
<html>
<head>
<title>Mehrwertsteuer</title>
<script language="JavaScript">
<!--
   function mwst()
   //hier wird die Mehrwertsteuer berechnet
    {
   var z=document.formular.Nettobetrag.value*16/100;
   z=Math.round(100*z)/100;
   document.formular.MWSt.value=z;
}
 //-->
</script>
</head>
<body>
<blockquote>
<body bgcolor=#ccFFFF>
<h1>Berechnung der Mehrwertsteuer</h1>
<form name="formular">
Nettobetrag : <input type=text name="Nettobetrag"><br>
<p><input type=button value="Berechnen" onClick="mwst()">
<p>Mehrwertsteuer: <input type=text name="MWSt">
</form>
</blockquote>
</body>
</html>
```

Im body-Teil wird ein Formular mit den beiden Feldern formular.Nettobetrag und formular.MWSt definiert. In der Berechnungsfunktion muss der Wert mit document.formular.Nettobetrag.value angesprochen werden. Das Arbeiten mit Formularwerten ist also ein wenig umständlicher als mit einfachen Variablen, aber der Aufwand lohnt sich. Man beachte, dass in der Funktion zunächst noch eine Hilfsvariable z eingeführt wird. Durch die Anweisung z=Math.round(100*z)/100; wird die Variable z, also die Mehrwertsteuer, auf zwei Nachkommastellen gerundet.

Math ist ein sog. eingebautes **Objekt** von Javascript, das verschiedene **Methoden** aus der Mathematik bereit stellt. Beispiele: a = Math.sin(x); b=Math.cos(x); c=Math.abs(x); d=Math.pow(basis,exponent);

Beispiel 5: Ein Euro-Rechner

Hier wird ein Formular mit dem Namen "rechner" definiert. Es hat zwei Felder für Ein- bzw. Ausgabe. Es sind dies "document.rechner.dm" und "document.rechner.euro". Zwei Buttons für die Umrechnung DM ---> EURO bzw. EURO ---> DM steuern den Aufruf der beiden Funktionen "euroberechnung()" und "dmberechnung()".
Die Größe der Eingabefelder kann dabei festgelegt werden, ebenso ein Vorgabewert.

```
<html>
<head>
<title>Euro-Rechner</title>
<script language="JavaScript">
<!--
function euroberechnung()
// hier wird mit den Formularwerten gerechnet und gerundet
{
var wert
 wert=document.rechner.dm.value/1.95583
// jetzt wird auf zwei Nachkommastellen gerundet.
 wert=Math.round(100*wert)/100;
 document.rechner.euro.value=wert;
}
function dmberechnung()
// hier werden Euro-Beträge in DM-Beträge umgerechnet.
{
var wert
 wert=document.rechner.euro.value*1.95583
 wert=Math.round(100*wert)/100;
 document.rechner.dm.value=wert;
}
//-->
</script>
</head>
<body>
<blockquote>
<body bgcolor=#CCFFFF>
<h1>EURO-Rechner</h1>
<form name="rechner">
<input type="text" name="dm" value="100" size="20"> <b>DM
<p></p>
<input type="text" name="euro" value="0" size="20"> EURO</b>
<p></p>
<input type="button"  value="DM--->Euro" onClick="euroberechnung()">
<input type="button"  value="Euro--->DM" onClick="dmberechnung()">
</form>
</blockquote>
</body>
</html>
```

5. Objekte

Objekte sind Datenstrukturen, die selbst Daten enthalten. Die Objekte enthalten **Eigenschaften** und **Methoden.**

Beispiele:

1. Das Objekt "**Navigator**" enthält u.a. die Eigenschaft "navigator.appName", also den offiziellen Namen des verwendeten Browsers.
2. Das Objekt "**Document**" enthält u.a. die Eigenschaften "document.bgColor", also die Hintergrundfarbe. Bereits verwendet wurde die Methode "**document.write()**", die Informationen ins aktuelle Dokument schreibt..
3. Das Objekt "**Window**" hat z.B. die Methoden "window.alert()" und "window.prompt()" zur Ausgabe oder Eingabe von Informationen. Die Methode "window.confirm(Mitteilung)" liefert ein Bestätigungsfenster und liefert die bool'schen Werte true oder false zurück.

Beispiel 6: Eigenschaften von Objekten

Das folgende kleine Programm liefert einige Informationen zum verwendeten Browser, zur Bildschirmauflösung und den Ort des aktuellen Dokumentes.

```
<html>
<head>
<title>Objekte in Javascript</title>
</head>
<body bgcolor=#88FFFF>
<script language="JavaScript">
<!--
document.write("<h4>Bildschirmeigenschaften</h4>");
document.write("<p>Höhe: ",screen.height);
document.write("<br>Breite: ",screen.width);
document.write("<br>Farbtiefe: ",screen.colorDepth);
document.write("<h4><p>Dokumenteigenschaften</h4>");
document.write("<p>Dokument-Farbe: ",document.bgColor);
document.write("<br>URL des Dokumentes: ",document.location.pathname);
document.write("<h4>Browsereigenschaften</h4>");
document.write("<p>Browsername: ",navigator.appName);
document.write("<br>Version: ",navigator.appVersion);
//-->
</script>
</body>
</html>
```

Auf dem Bildschirm erscheinen dann Informationen in der folgenden Art:

Bildschirmeigenschaften

Breite: 800
Höhe: 600
Farbtiefe: 16

6. Bedingte Anweisungen

Als bedingten Anweisung kennt JavaScript die if-Anweisung.

```
if ( Bedingung)
Anweisung1;
else
Anweisung2;
```

Hat die Bedingung den Wahrheitswert true wird die Anweisung1 ausgeführt, ansonsten die Anweisung2. Der else Teil kann fehlen. Anstelle einer einzigen Anweisung kann auch ein Anweisungsblock stehen, der mit { und } geklammert wird. Folgendes Beispiel ermittelt das Maximum zweier Zahlen:

```
if (n>=m)
max = n;
else
max = m;
```

Mögliche Bedingungen (Vergleichsoperatoren) sind:

gleich (a==b),
kleiner (a<b),
kleiner oder gleich (a<=b),
größer (a>b),
größer oder gleich (a>=b),
ungleich (a!=b).

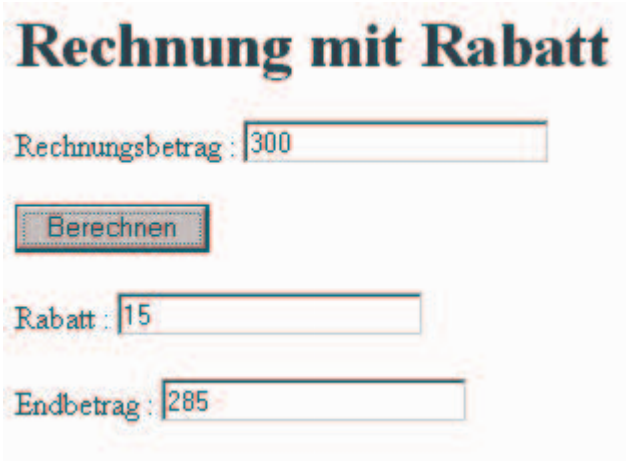
Um Abfragen komplexer gestalten zu können werden logische Operatoren verwendet.
&& bedeutet **logisch und**
| | bedeutet **logisch oder** *(auf der Tastatur neben dem <-Zeichen)*
! bedeutet **nicht** (Umkehrung)

Einfaches Beispiel: Test, ob drei Zahlen gleich sind

```
<script>
<!--
var text;
var a=7;
var b=7;
var c=7;
if(a==b && a==c)
{
  text="Alle Werte sind gleich!";
  document.write(text);
}
//-->
</script>
```

Beispiel 7: Es soll ein Formular entworfen werden, das ein Eingabefeld für einen Rechnungsbetrag enthält. Falls dieser größer als 100 ist, so wird ein Rabatt in Höhe von 5 % gewährt. Der Rabatt und der zu zahlende Betrag sollen ebenfalls in das Formular geschrieben werden.

```
<html>
<head>
<title>Mehrwertsteuer</title>
<script language="JavaScript">
<!--
   function rechnen()
{
   var z = document.formular.Rechnungsbetrag.value;
   if ( z > 100)
   {document.formular.Rabatt.value = z*0.05}
   else
   {document.formular.Rabatt.value = 0;}
   document.formular.Endbetrag.value = z - document.formular.Rabatt.value;
 }
 //-->
</script>
</head>
<body>
<body bgcolor=#ccFFFF>
<h1>Rechnung mit Rabatt</h1>
<form name="formular">
Rechnungsbetrag : <input type=text name="Rechnungsbetrag"><p>
<input type=button value="Berechnen" onClick="rechnen()"><p>
Rabatt : <input type=text name="Rabatt"><p>
Endbetrag : <input type=text name="Endbetrag"><p>
</form>
</body>
</html>
```



So sieht's auf dem Bildschirm aus.

7. Die Zählschleife

Schleifen dienen dazu, Anweisungen mehrmals durchzuführen. Ist bekannt, wie oft die Anweisungen wiederholt werden sollen, so bietet sich die Zählschleife an. Sie hat die Syntax:

for(Startwert; Endbedingung; Erhöhung der Zählvariablen)
{ Anweisungen }

Beispiel: **for (n = 1 ; n <= 10 ; n = n + 1)**
{document.write("
", n*n)

Diese Schleife schreibt die ersten 10 Quadratzahlen auf den Bildschirm.

Beispiel 8: Wertetabelle einer Funktion

```
<html><head>
<title>Javascript</title>
</head>
<blockquote>
<body bgcolor=#CCFFFF>
<h1>Schleife für Wertetabelle</h1></h1>
<script language="JavaScript">
<!--
function f(x)
{
var hilf=x*x-4;
return Math.round(1000*hilf)/1000
}

var x,y;
for(x = -5 ; x <= 5 ; x = x + 1 )
{ y=f(x);
  document.write("<br> x = ", x,"    y = ", y);
}
//-->
</script>
</blockquote>
</body>
</html>
```

x = -5 y = 21
x = -4 y = 12
x = -3 y = 5
x = -2 y = 0
x = -1 y = -3
x = 0 y = -4
x = 1 y = -3
x = 2 y = 0
x = 3 y = 5
x = 4 y = 12
x = 5 y = 21

Zunächst wird eine mathematische Funktion f mit einem Eingabewert x definiert. In der Zählschleife läuft die Variable x dann von von –5 bis 5 mit der Schrittweite 1.

8. Die While-Schleife

Oft ist nicht bekannt, wie oft eine Schleife durchlaufen werden muss. Dann kann die While-Wiederholung verwendet werden. Sie hat die Syntax:

while (Bedingung)
{Anweisungen}

Beispiel 9: Ein kleines Zahlenrätsel

Der Anwender muss eine Zahl erraten. Dies wird so lange wiederholt, bis die richtige Zahl gefunden ist.

```
<html>
<head>
<title>Zahlenrätsel</title>
<script language="JavaScript">
<!--
var zahl=0;   //Startwert
var loesung = 100*Math.random();
loesung=Math.floor(loesung)+1;
while (zahl!=loesung)
{
zahl=prompt("Bitte geben Sie eine Zahl bis 100 ein!","");
if (zahl<loesung)
{alert("Zahl zu klein");}
if (zahl > loesung)
{alert("Zahl zu groß");}
}
document.bgColor="red";
//-->
</script>
</head>
<body>
<h4>Prima, geschafft!</h4>
</body>
</html>
```

Für die Formulierung der Abbruchbedingung stehen die Vergleichsoperatoren ==, <, <=, >, >= und != zur Verfügung. Außerdem wird eine **bedingte Anweisung** verwendet, um darüber zu informieren, ob die geratene Zahl zu groß oder zu klein war. Wichtig ist es, vor der while-Anweisung einen **Startwert** sinnvoll zu setzen.

Die Anweisung "var loesung = 100*Math.random();" erzeugt eine Zufallszahl von 0 bis 100, Math.floor() liefert den ganzen Anteil dieser Zahl. Damit die 0 nicht entsteht, wird noch 1 addiert.

9. Funktionen

Soll der Befehlsumfang von Javascript erweitert werden, so können eigene Funktionen geschrieben werden. Funktionen vereinfachen häufig benötigte Berechnungen und machen Programme übersichtlicher.
Eine Funktionsdefinition beginnt mit dem Schlüsselwort **function,** gefolgt vom Namen der Funktion und evt. einer Liste von Eingabe-Parametern in runden Klammern eingeschlossen. Fehlen diese Parameter, müssen die Klammern dennoch stehen bleiben.

Beispiel 10: Funktion zum Drucken eines Textes in roter Schrift

```
<html>
<head>
<title>Javascript</title>
<script language="JavaScript">
<!--
function printred(text)
{  document.write("<font color =red>",text,"</font>");
}
printred("Meine 1. Funktion!");
//-->
</script>
</head>
</html>
```

Der Aufruf einer Funktion kann einfach über ihren Namen geschehen. Printred(...) stellt den übergebenden Text rot dar.
Falls Funktionen einen Wert zurückgeben, so kann dies mit return geschehen.

Beispiel: Die Funktion *quadervol (l , b , h)* liefert als Funktionswert das Volumen eines Quaders mit den Kantenlängen l, b, und h zurück.

```
function quadervol(l,b,h)
// berechnet das Volumen eines Quaders
{  var hilf;
   hilf = l*b*h;
   return hilf;   }
```

Aufrufbeispiel: `document.write("Volumen = ",quadervol(3,4,5));`

Bez.: Die Variable hilf ist eine **lokale Variable**. Sie wird innerhalb der Funktion deklariert und ist nur dort bekannt. Eine **globale Variable** ist dagegen im Hauptprogramm deklariert und überall bekannt.

Bekannt ist der Funktionsbegriff auch aus der Mathematik. Eine reelle Funktion f bestimmt zu einem x-Wert einen Funktionswert f(x).

```
function f(x)
{  var y = x*x - 2;
   return y          }
```

Beispiel 11: **Systematische Suche nach Primzahlen**

Im folgenden Beispiel wird zunächst eine Funktion *teilerzahl(n)* definiert, die als Funktionswert die Anzahl der Teiler der natürlichen Zahl n liefert. Ist diese Zahl 2, so handelt es sich um eine Primzahl. In einer anschließenden for-Schleife werden mit dieser Methode systematisch alle Primzahlen von 1 bis 100 ermittelt.

```
<html>
<head>
<script language="JavaScript">
<!--
function teilerzahl(n)
// berechnet die Anzahl der Teiler von n
{  var hilf = 0;
   for (k=1 ; k <= n; k = k + 1)
      if (n % k == 0) hilf = hilf +1;
   return hilf;
}

document.write("<h1> Primzahlen bis 100</h1>");
document.write("<blockquote>");
for (n = 1; n <= 100; n = n + 1)
   if (teilerzahl(n) == 2) document.write("<br>",n);
//-->
</script>
</head>
</html>
```

Das Programm liefert tatsächlich alle Primzahlen von 2 bis 100.

2 3 5 7 11 13 17 19 23 29 31 37 41 43
47 53 59 61 67 71 73 79 83 89 97

Beispiel 12: **Berechnung von Fakultäten**

In der Wahrscheinlichkeitsrechnung spielt die sog. **Fakultät** einer natürlichen Zahl eine wichtige Rolle. Die Fakultät einer Zahl n ist das Produkt 1*2*3*...*n.
Speziell gilt: 0! = 1 und 1! = 1.

```
function fak(n)
// berechnet die Fakultät von n
{ var hilf = 1;
if (n==1 || n==0) return 1
  else
  for (k=1 ; k <= n; k = k + 1)
     hilf = hilf * k;
     return hilf  }
```

10. Rekursive Funktionen

Die Definition eines Problems, eines Algorithmus, einer Funktion oder einer Datenstruktur durch sich selbst heißt ***Rekursion***. Funktionen können sich selbst aufrufen. Sie heißen dann *rekursiv*.

Arten der Rekursion

a) ***Lineare Rekursion****:* Eine rekursive Funktion f heißt *linear rekursiv*, wenn eine Aufruf von f jeweils maximal einen weiteren Aufruf von f nach sich zieht.

Beispiel 13: die Fakultät einer Zahl - rekursiv

Offenbar gilt für die Fakultät einer Zahl: n! = n*(n-1)!

Diese rekursive Definition kann unmittelbar in Javascript übersetzt werden.

```
<script language="JavaScript">
<!--
   function fak(n)
   //hier wird rekursiv gerechnet
    {
   if (n==0)   // Rekursionsanker
return 1
else
return fak(n-1)*n
}   //Rekursionsschritt
//-->
</script>
```

Ein möglicher Aufruf ist : document.write("<h3>","Fakultät von ",5," ist : ",fak(5),"</h3>");

b) ***Baumartige Rekursion****:* Eine rekursive Funktion f heißt *baumartig rekursiv*, wenn es einen Aufruf von f gibt der mindestens zwei weitere Aufrufe von f nach sich zieht.

Beispiel 14: Die Fibonacci-Funktion

Die Fibonacci-Funktion ist folgendermaßen definiert:

fib(0) = fib(1) = 1
fib(n) = fib(n-1)+fib(n-2) falls n>2 ist.

Lösung in Javascript

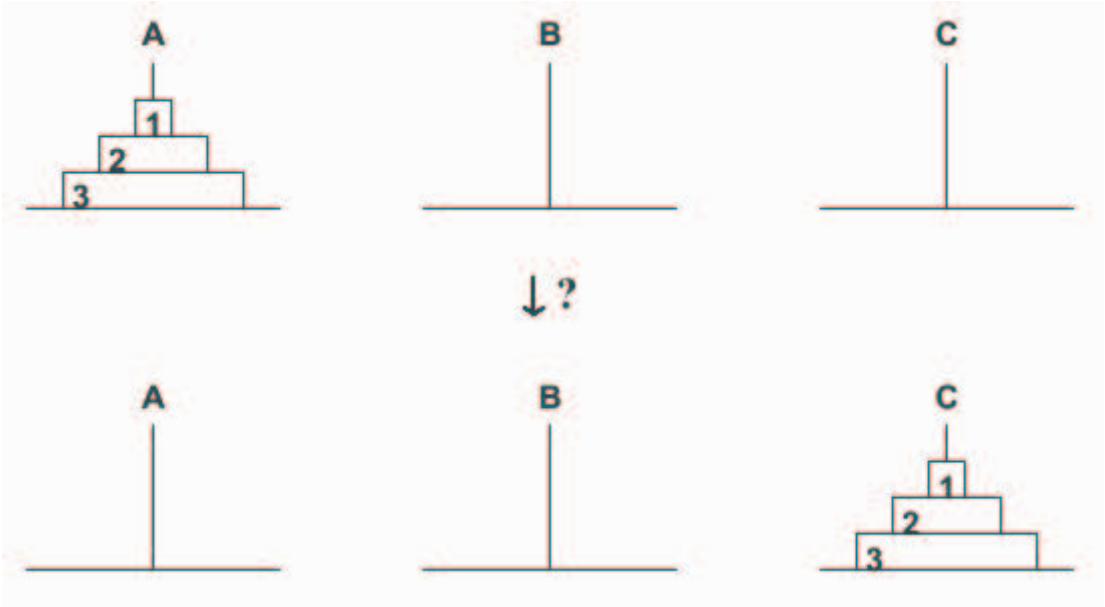
```
<html>
<head>
<title>Fibonacci-Zahlen rekursiv</title>
</head>
<body>
<blockquote>
<body bgcolor=#00FFFF>
<h1>Berechnung der Fibonacci-Zahlen</h1>

<script language="JavaScript">
<!--
   function fib(n)
   //hier wird rekursiv gerechnet
    {

   if (n==1 || n==0)
return 1
else
  return fib(n-1)+fib(n-2)
}
for ( n = 1 ; n <= 15; n = n + 1)
document.write("Fib( ",n,") ist : ",fib(n),"<br>")

//-->
</script>
</blockquote>
</body>
</html>
```

Das Programm liefert die ersten 15 Fibonacci-Zahlen:

```
Fib( 1) ist : 1
Fib( 2) ist : 2
Fib( 3) ist : 3
Fib( 4) ist : 5
Fib( 5) ist : 8
Fib( 6) ist : 13
Fib( 7) ist : 21
Fib( 8) ist : 34
Fib( 9) ist : 55
Fib( 10) ist : 89
Fib( 11) ist : 144
Fib( 12) ist : 233
Fib( 13) ist : 377
Fib( 14) ist : 610
Fib( 15) ist : 987
```

<u>Beispiel 15:</u> Die Türme von Hanoi

Beim Spiel ”Türme von Hanoi“ geht es darum, drei gelochte Scheiben unterschiedlicher Größe die auf einem Stab A in absteigender Größe aufeinandergelegt sind, unter Beachtung gewisser Regeln auf einen weiteren Stab C umzulegen. Dafür darf ein Hilfsstab B benutzt werden. Start- und Zielzustand sind folgendermaßen:



Dies sind die Regeln:

- Nur eine Scheibe auf einmal bewegen, nämlich die oberste eines beliebig gewählten Stapels!

- Niemals eine größere auf eine kleinere Scheibe legen!

Lösung für 3 Scheiben:

Scheibe von A ----> C
Scheibe von A ----> B
Scheibe von C ----> B
Scheibe von A ----> C
Scheibe von B ----> A
Scheibe von B ----> C
Scheibe von A ----> C

Die rekursive Lösung ist überraschend einfach. Sie lautet:

- Wenn nur eine Scheibe auf einem Stab liegt, so lege diese auf den Zielstab.
- Liegen n Scheiben (n>1) auf einem Stab, so verschiebe die obersten n-1 Scheiben auf den Hilfsstab, lege die unterste Scheibe auf den Zielstab, verschiebe die n-1 Scheiben vom Hilfsstab zum Zielstab.

Diese Lösung lässt sich sofort in ein Programm umsetzen. Dazu wird eine Funktion **verschiebe(n,A,B,C)** definiert, die n Scheiben von A nach C unter Verwendung des Hilfsstabes B verschieben soll.

```
<html>
<head>
<title>Türme von Hanoi</title>
</head>
<body>
<blockquote>
<body bgcolor=BBFFFF>
<h1>Türme von Hanoi</h1>
<script language="JavaScript">
<!--
function verschiebe(n, A, B, C)
// A = Start, B = Hilfsstab, C = Ziel
{
if (n == 1)
document.write("<br> Scheibe von ",A," ----> ",C)
else
{
verschiebe(n – 1 ,   A , C , B);
document.write("<br> Scheibe von ",A," ----> ",C)
verschiebe(n – 1 ,   B ,   A ,  C);
}
}
var anzahl = prompt("Gib die Scheibenzahl an (1 bis 7) : ",3)
document.write("Lösung für ", anzahl , " Scheiben: <br>");
// Aufruf der rekursiven Funktion verschiebe (...)
verschiebe(anzahl, "A" , "B" , "C");
//-->
</script>
</blockquote>
</body>
</html>
```

Anm.: Alternativ kann man die rekursive Funktion auch folgendermaßen formulieren.

```
function verschiebe(n, A, B, C)
// A = Start, B = Hilfsstab, C = Ziel
{
  if (n > 0)
  {
    verschiebe(n - 1, A , C , B);
    document.write("<br> Scheibe von ",A," ----> ",C)
    verschiebe(n - 1, B, A, C);
  }
}
```

12. Interaktive Gestaltung von Internetseiten

Eine der wichtigsten Anwendungen für Javascript-Programme ist die interaktive Gestaltung von Internetseiten. Die folgenden Beispiele sollen dazu einige Anregungen geben.

Beispiel 16: Aufklappen eines Hinweisfenster per Mausklick

```
<html>
<head>
<title>Aufklappen eines Hinweisfensters</title>
<SCRIPT>
function Fenstermeldung() {
alert("Skripte und Anleitungen zu HTML und Javascript gibt es auf der Homepage
http://bildung.freepage.de/gsginf11 zu Herunterladen.");
}
</SCRIPT>
</head>
<body bgcolor=77ffff>
<form>
<input type="button" value="Weitere Informationen per Klick!"
 onClick=Fenstermeldung()>
</form>
</body>
</html>
```

Beispiel 17: Aufklappen von Eingabefenstern

```
<html>
<head>
<title>Aufklappen von Eingabefenstern</title>
<SCRIPT>
function EingabeTel () {
var telnummer=prompt("Bitte geben Sie Ihre Telefonnumer ein - wir rufen Sie
zurück!",0 );
}
function EingabeAdress () {
var adresse=prompt("Bitte geben Sie Ihre Adresse ein - wir schicken Ihnen die
Unterlagen zu!","Lebach");
}
</SCRIPT>
</head>
<body>
<form>
<input type="button" value="Sie interessieren sich für Lebensversicherungen?!"
onClick=EingabeTel()>
<P>
<input type="button" value="Sie interessieren sich für Hausratsversicherungen?!"
onClick=EingabeAdress()>
</form>
</body>
</html>
```

Beispiel 18: Überprüfung von Eingabefeldern

Im folgenden Beispiel wird überprüft, ob überhaupt ein Text in ein Eingabefeld eingegeben wurde. Die Eingabefelder werden jeweils in einem Formular definiert.

```
<html>
<head>
<title>Prüfen, ob Texteingabe erfolgte</title>
<script language="JavaScript">
function test(eingabe)
{
if (eingabe == "")
alert("Hinweis: Eingabe unvollständig: Es wurde nichts eingegeben!")
else
alert("Hinweis: Die Eingabe ist erfolgt!");
}
</script>
</head>
<body>
<form name="formular">
<INPUT NAME ="nachname" VALUE="" SIZE=40 MAXLENGTH=40>
<input TYPE="button" VALUE="Test der Eingabe?"
OnClick="test(formular.nachname.value)">
</form>
<INPUT type="button" value="Oder einfach eine Seite zurück... Bitte anklicken!"
onclick="history.go(-1)"></FORM></P></CENTER>
</body>
</html>
```

Bemerkenswert ist die Zeile:

```
<INPUT type="button" value="Oder einfach eine Seite zurück... Bitte anklicken!" onclick="history.back()"></FORM></P>
```

Mit diesem Befehl springt man automatisch zur letzten Seite zurück. Diese Zeile sollte auf keiner Internetseite und in keiner Navigationsleiste fehlen! Script-Tags sind hierzu nicht erforderlich.

Übungsbeispiel: **Idealgewicht und Body-mass-Index**

Es wird ein Formular definiert, das zwei Eingabefelder und zwei Ausgabefelder hat. Die Größe des Ausgabefeldes kann bei der Definition festgelegt werden, ebenso kann man einen Startwert vorgeben.

```
<html>
<head>
<title>Gewicht</title>
<script language="JavaScript">
<!--
function berechnung()
//ab hier wird mit den Formularwerten gerechnet und gerundet
{
var wert
wert=document.rechner.laenge.value-100
document.rechner.normgewicht.value=wert;
wert=document.rechner.gewicht.value/(document.rechner.laenge.value*
document.rechner.laenge.value)*10000;
wert=Math.round(10*wert)/10;
document.rechner.bmindex.value=wert;
}
//-->
</script>
</head>
<body>
<blockquote>
<body bgcolor=#CCFFFF>
<h1>Idealgewicht und Body-Mass-Index</h1>
<form name="rechner">
<input type="text" name="gewicht" value="75" size="10"> <b>DM
<p></p>
<input type="text" name="laenge" value="176" size="10"> Länge in cm</b>
 <p></p><input type="button"  value= "Berechnen" onClick="berechnung()">
<p><hr><b>
<p>/<p>
<input type="text" name="bmindex" value="0" size="10"> Body-Mass-Index
<p></p>
<input type="text" name="normgewicht" value="0" size="10"> Normgewicht
</form>
</blockquote>
</body>
</html>
```