JavaScript

- Einführende Beispiele

http://pcnews.at/edu/tk/javascr/~javascr.htm
ftp://pcnews.at/dsk/5x/54x/541/javascr/bsp*.htm/

Franz Fiala

Jedes HTML-Dokument kann sowohl als eine Folge von HTML-Tags und Text als auch als eine Folge von Skript-Anweisungen geschrieben werden. Als Skriptsprachen können

- Javascript (Netscape) oder
- Visual-Basic-Script (Microsoft)

verwendet werden. JavaScript ähnelt in seinem Aufbau der Sprache C/C++, VB-Script ist verwandt zu Visual-Basic und zu Visual Basic for Applications, das man im Office-Paket von Microsoft mitgeliefert bekommt.

JavaScript ist in Explorer und Navigator eingebaut, wenn auch geringe Implementierungsunterschiede bestehen - aber das ist man von HTML ja gewöhnt.

Mit Javascript erhält man einerseits eine Computersprache, die mit den üblichen Kommadowörtern zur Steuerung des Ablaufs ausgestattet ist und anderseits eine Klassenbibliothek, die den Zugrif auf die verschiedenen Elemente einer HTML-Seite erlaubt.

Mit einem solchen Objekt sind verbunden:

- Eigenschaften (Properties)
- Methoden (Methods), erkennbar an der folgenden runden Klammer, entspricht einer Funktion in anderen Sprachen und
- Event Handler (Ereigisprozeduren), die im Zuge von Benutzeraktivitäten ausgelöst werden (Mausbewegung oder Mausklick oder Tastendruck)
- Welche Objekte verfügbar sind, entnehmen Sie bitte der folgenden Referenz.

document

Ein wichtiges Objekt ist das aktuelle Dokument, dessen bekannte Attribute wie Hintergrundfarbe, Textfarbe und andere sofort als Properties zugeordnet werden können.

Die Methoden des Dokuments sind `open()`, `close()` oder `write()`.

Eigenschaften sind Farbgebung, URL, Zeitstempel und andere.

```
<HTML>
  <HEAD>
    <TITLE>BSP01: Attribute statisch</TITLE>
  </HEAD>
  <BODY BGCOLOR="#800000">
    Gewöhnliches HTML-Dokument
  </BODY>
</HTML>
```

Man kann aber dasselbe Dokument auch durch ein Skript erstellen.

Ei
n Skript entsteht durch den Tag <SCRIPT> und kann entweder im HEAD-Abschnitt oder im BODY-Abschnitt stehen.

Skripts im HEAD-Abschnitt sind im allgemeinen Funktionen und Variablen. Sie haben beim Laden des Dokuments noch keine Wirkung. Sie wirken erst, wenn eine Aktion im HTML-Kode - eine Ereignisfunktion - sie aktiviert.

Skripts im BODY-Abschnitt werden im Zuge des Aufbaus der HTML-Seite ausgeführt.

Wenn wir in unserem Beispiel im Zuge des Aufbaus der Seite Skripts einbauen, können wir beispielsweise den Titel oder die Textfarbe verändern.

Eigenschaften von Objekten

Objekt.Eigenschaft=Wert
Document.bgColor="#800000"

```
<HTML>
  <HEAD>
    <TITLE>BSP02: Attribute dynamisch</TITLE>
  </HEAD>
  <BODY BGCOLOR="#808080">
    <SCRIPT>
      document.bgColor="#FFFFFF"
    </SCRIPT>
    Gewöhnliches HTML-Dokument
  </BODY>
</HTML>
```

Hier wurde - noch vor dem Schreiben der Zeile "Gewöhnliches HTML-Dokument" der Hintergrund von dokument auf weiß statt grau geändert.

Methoden von Objekten

Objekt.Methode(Parameter)
document.write(„Text, geschrieben durch Script")

```
  <HEAD>
    <TITLE>BSP03: Textausgaben</TITLE>
    <SCRIPT>
      function Textausgabe(text)
      {
        document.write(text)
      }
    </SCRIPT>
  </HEAD>
  <BODY BGCOLOR="#FFFFFF">
    Gewöhnlicher Text<BR>
    <SCRIPT><!-- Texteingabe mit Objekt-Methode-->
      document.fgColor="#008000"
      document.write("dynamisch generierter Text mit
write()<BR>")
      Textausgabe("Textausgabe mit Funktion<BR>")
    </SCRIPT>
  </BODY>
</HTML>
```

Es erscheint nicht sehr wichtig, daß man Eingaben auf diese Weise indirekt ausführt, doch kann man mit diesem Hilfsmittel auch Dinge anzeigen, die man sonst nur manuell, nicht aber automatisch generieren kann, etwa das Datum und die Uhrzeit des Speicherdatums:

```
<HTML>
  <HEAD>
    <TITLE>BSP04: Speicherdatum</TITLE>
  </HEAD>
  <BODY BGCOLOR="#808080">
```

```
    <SCRIPT>
      document.write(document.lastModified)
    </SCRIPT>
  </BODY>
</HTML>
```

Oder den URL des Dokuments

```
<HTML>
  <HEAD>
    <TITLE>BSP05 Dokument-URL</TITLE>
  </HEAD>
  <BODY BGCOLOR="#808080">
    <SCRIPT>
      document.writeln(document.referrer)
    </SCRIPT>
  </BODY>
</HTML>
```

window

Während document das aktuelle Dokument meint, kann das übergeordnete Objekt window mit beliebigen Fenstern umgehen.

```
<HTML>
  <HEAD>
    <TITLE>BSP 06: window</TITLE>
  </HEAD>
  <BODY>
    <SCRIPT>
      window.open(„bsp05.htm“,"neuer Titel",
              „toolbar=false, location=false, “
            + „directories=false, status=false, “
            + „scrollbars=false, resizable=true, “
            + „width=200, height=200"

      window.status="\Achtung-Fenster wird geöffnet"

      document.write(„Achtung-Fenster wird geöffnet<BR>“)
      window.alert(„Achtung“)

      document.write(„Alternative<BR>“)
      if (window.confirm(„Sind Sie sicher?“))
      {
        alert („Ja“)
      }
      else
      {
        alert („Nein“)
      }

      document.write(„Frage mit Vorgabewert<BR>“)
      window.prompt(„Sind Sie sicher?“,"Ja")

    </SCRIPT>
  </BODY>
</HTML>
```

- In diesem Dokument gibt es einige Besonderheiten:
- Man kann mit open() beliebige weitere Fenster eröffnen; mehr noch: man kann die Fensterparameter gezielt steuern:

toolbar	Werkzeugleiste
location	URL-Leiste
directories	Tools und Verzeichnisse
status	Statuszeile
scrollbars	Laufleisten anzeigen
resizable	Fenster in Größe veränderbar
width, height	Dimensionen in Pixel

Zeichenfolgen (Strings) können mit dem Plus-Operator verkettet werden.

- Reihenfolge der Programmzeilen entspricht nicht dem tatsächlichen Ablauf.

location

Das location-Objekt liefert Informationen über das aktuelle Dokument. Es besitzt keine Methoden oder Ereignisfunktionen. Das folgende Beispiel demonstriert alle verfügbaren Eigenschaften.

```
<HTML>
  <HEAD>
    <TITLE>BSP07: location</TITLE>
  </HEAD>
  <BODY>
    <SCRIPT>
      document.write(„<PRE>“)
      document.write(„hash    :“ + location.hash + „<BR>“)
      document.write(„host    :“ + location.host + „<BR>“)
      document.write(„hostname:“ + location.hostname +
„<BR>“)
      document.write(„href    :“ + location.href + „<BR>“)
      document.write(„pathname:“ + location.pathname +
„<BR>“)
      document.write(„port    :“ + location.port + „<BR>“)
      document.write(„protocol:“ + location.protocol +
„<BR>“)
      document.write(„search  :“ + location.search +
„<BR>“)
      document.write(„</PRE>“)
    </SCRIPT>
  </BODY>
</HTML>
```

Interaktive Dokumente

Die bisherigen Beispiele fügten einen SCRIPT-Abschnitt im BODY-Abschnitt ein und die Script-Anweisungen wurden an dieser Stelle ausgeführt. Für Interaktivität ist es aber erforderlich, Aktivitäten in Abhängigkeit von Benutzereingaben zu steuern. Grundsätzlich sind dazu alle Formulartags INPUT, TEXT und SELECT geeignet. Wir beschränken uns hier auf das INPUT-Tag in der Variante BUTTON.

In einem Dokument sind beliebig viele Formularabschnitte einbaubar. Anders als bei Formularen, deren Ziel eine Datenbank am Server ist (serverseitige CGI-Scripts), benötigt man bei Formulartags, die lediglich eine clientseitige Aktivität auslösen kein ACTION und kein METHOD-Attribut im FORM-Tag.

```
<HTML>
  <HEAD>
    <TITLE>BSP08: Input-Buttons</TITLE>
  </HEAD>
  <BODY>
    <FORM>
       <INPUT TYPE="BUTTON" VALUE="Button 1"><BR>
    </FORM>
    <FORM>
       <INPUT TYPE="BUTTON" VALUE="Button 2"><BR>
    </FORM>
  </BODY>
</HTML>
```

Das wichtigste zusätzliche Element ist eine Ereignisprozedur, die immer dann aufgerufen wird, wenn der Benutzer eine bestimmte Aktion setzt. Bei einem BUTTON ist es die Prozedur onClick(). Im einfachsten Fall kann die gewünschte Aktion durch “” eingegrenzt werden.

```
  <HEAD>
    <TITLE>BSP10: Hintergrundfarbe ändern</TITLE>
  </HEAD>
  <BODY>
    <FORM>
       <INPUT TYPE="BUTTON"
              VALUE="Hintergrundfarbe rot"
              onClick="document.bgColor='red'"><BR>
```

```
    </FORM>
    <FORM>
       <INPUT TYPE="BUTTON"
              VALUE="Hintergrundfarbe blau"
              onClick="document.bgColor='#0000FF'"><BR>
    </FORM>
  </BODY>
</HTML>
```

Bei einfachen Zuweisungen von Objekteigenschaften kann man - wie im oberen Beispiel - diese Zuweisung gleich als einen Kodeabschnitt im Rahmen des INPUT-Tag vorsehen (Inline-Kode). Bei komplexeren Aufgaben, wird eine eigene Funktion geschrieben.

Diese Prozeduren sind Kodeabschnitte, die zunächst nur geladen, und nur bei einem Mausklick des Benutzers ausgelöst werden. Ihr Platz ist nicht der BODY- sondern der HEAD-Abschnitt.

```
<HTML>
  <HEAD>
    <TITLE>BSP09: Hintergrundfarbe ändern</TITLE>

    <SCRIPT>
      function Hintergrund(Farbe)
      {
        document.bgColor=Farbe
      }
    </SCRIPT>

  </HEAD>
  <BODY>
    <FORM>
       <INPUT TYPE="BUTTON"
              VALUE="Hintergrundfarbe rot"
              onClick="Hintergrund('red')"><BR>
    </FORM>
    <FORM>
       <INPUT TYPE="BUTTON"
              VALUE="Hintergrundfarbe blau"
              onClick="Hintergrund('#0000FF')"><BR>
    </FORM>
  </BODY>
</HTML>
```

Die Funktion

- beginnt mit dem Schlüsselwort function, gefolgt von dem Funktionsnamen und einem Klammernpaar.
- Die Funktionsanweisungen sind mit mit geschwungenen Klammern eingeschlossen
- in der Funktionsklammer steh(en) Parameter oder nichts (parameterlose Funktion

Im Beispiel wird die Hintergrundfarbe als Parameter Farbe übergeben, wobei die Farbe sowohl als rgb-Wert als auch als reserviertes Wort für die Farbe übergeben werden kann.

links

Beliebter Ort zur Einblendung von Hinweisen für Benutzer ist die Statuszeile, die mit window.status=text vorgenommern werden kann. Eine Möglichkeit ist eine nähere Erklärung zum jeweiligen URL.

```
<HTML>
  <HEAD>
    <TITLE>BSP 11: Statuszeile</TITLE>
  </HEAD>
  <BODY>
    Üblicherweise wird in der
    <A HREF="URL1"
       onMouseOver="window.status='Die Statuszeile '
                    + 'ist für Erklärungen gut geeignet'">
    Statuszeile</A> der URL des
    <A HREF="URL2"
       onMouseOver=
         „window.status='Verzweigt zu einer anderen
Seite'“>
    Hyperlinks</A> angegeben.
  </BODY>
</HTML>
```

Scrollende Mitteilung

Laufende Texte können unter Zuhilfenahme der Timeout-Methode erzeugt werden.

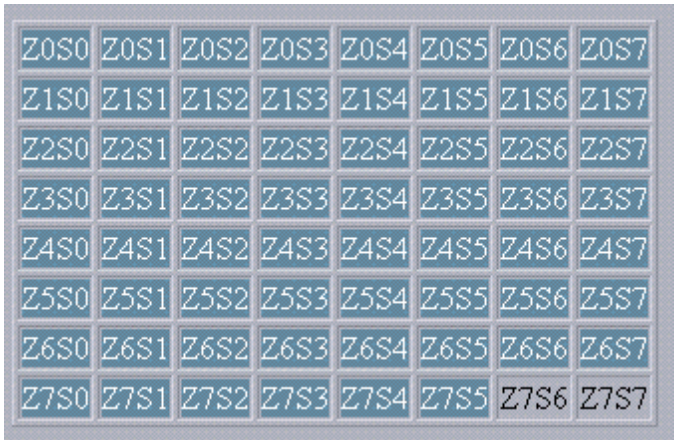
```
<HTML>
  <HEAD>
    <TITLE>BSP 12: Scrollender Statuszeilentext</TITLE>
    <SCRIPT>
    var Meldung="Bewegung erhöht die Aufmerksamkeit  ***
"
    var Verzoegerung=200
    var timerId
    function Laufschrift()
    {
      Meldung = Meldung.substring(1,Meldung.length)+
                Meldung.substring(0,1)
      timerId = setTimeout(„Laufschrift()“,Verzoegerung)
      window.status=Meldung
    }
    </SCRIPT>
  </HEAD>
  <BODY onLoad="Laufschrift()">
  </BODY>
</HTML>
```

Steuerstrukturen

Die Steuerstrukturen sind der Sprache C praktisch ident. Im folgenden Beispiel wird gezeigt, wie man eine aufwendigere Tabelle statt durch Aneinanderreihen von HTML-TAGs durch Wiederholung in einer for-Schleife erzeugt werden kann.

```
<HTML>
  <HEAD>
    <TITLE>BSP13: Steuerstrukturen</TITLE>
  </HEAD>
  <BODY>
    <TABLE BORDER=1>
    <SCRIPT>
    for (Zeile=0; Zeile<8; Zeile++)
    {
      document.write(„<TR>“)
      for (Spalte=0; Spalte<8; Spalte++)
      {
        document.write(„<TD>Z“+Zeile+"S"+Spalte+"</TD>")
      }
      document.write(„</TR>“)
    }
    </SCRIPT>
    </TABLE>
  </BODY>
</HTML>
```

Das entstandene Bild:



Eingaben und Berechnungen

Als Eingabeobjekt werden INPUT-Formulare des Typs TEXT verwendet. Alle Eingaben, auch wenn es sich um Zahlen handelt, sind Strings und müssen im Zuge von Berechnungen

mit Hilfe der String-Methoden `parseInt` oder `parseFloat` in Zahlen umgewandelt werden. Nach Durchführung der Berechnung erfolgt die Ausgabe wieder in einem TEXT-Feld, daher muß die entstandene Zahl wieder in einen String verwandelt werden. Das geschieht durch Addition mit einem String:

String = "" + 5


```
<HTML>
  <HEAD>
    <TITLE>BSP 14: Eingaben</TITLE>
  </HEAD>
  <BODY>
    <H2>Rechnen</H2>
    <FORM>
      X
      <INPUT TYPE="TEXT"
             NAME="X"
             VALUE=2>
      + Y
      <INPUT TYPE="TEXT"
             NAME="Y"
             VALUE=3>
      <INPUT TYPE="BUTTON"
             VALUE="="
             onClick= 'this.form.Z.value= „“ +
                      (parseFloat(this.form.X.value) +
                      parseFloat(this.form.Y.value))' >
      Z
      <INPUT TYPE="TEXT"
             NAME="Z"
             VALUE=0>
    </FORM>
  </BODY>
</HTML>
```

Diese Einführung ist kein Ersatz für eine eingehende Beschäftigung mit weiterführenden Materialien, wie z.B.: JavaScript Handbook, Danny Goodman, IDG Books, ISBN 0-7645-3003-8, inklusive CD.
Referenz (http://pcnews.at/edu/tk/javascr/ref/~ref.htm)

Beispiele (http://pcnews.at/edu/tk/javascr/bsp/~bsp.htm)
Beachten Sie auch die Kurzreferenz nach dieser Einführung

Javascript

- Kurzreferenz

Franz Fiala

Eingebaute Objekte

```
window      history
     location
     document
             link
             anchor
             form
                     text, textarea, password
                     radio
                     button, reset, submit
                     checkbox
                     select
string
Date
Math
navigator
```

Windows & Frames

Creating Windows

```
windowObject = window.open([parameters])
<BODY
     ...
     [onLoad="handlerTextOrFunction"]
     [onUnload="handlerTextOrFunction"]>
</BODY>
```

Creating Frames

```
<FRAMESET>
     ROWS="ValueList"
     COLS="ValueList"
          onLoad="handlerTextOrFunction"]
          onUnload="handlerTextOrFunction"]>
     <FRAME SRC="locationOrURL" NAME="
firstFrameName">
     ...
     <FRAME SRC="locationOrURL" NAME="lastFrameName
„>
</FRAMESET>
```

Properties

```
status                    (string)
defaultStatus             (string)
frames                    (array)
parent                    (Window object)
Self                      (Window object)
top                       (Window object)
window                    (Window object)

alert(message )
confirm(message )
prompt(message, defaultReply )
open(„URL“, „windowName“ [, „windowFeatures“])
close()
setTimeout(„expression“, millisecondsDelay )
clearTimeout(timeoutIDnumber )
```

Event Handlers

```
onLoad=
onUnload=
```

Location Object

Properties

```
href (string)
hash (string)
host (string)
hostname     (string)
pathname     (string)
port (string)
protocol     (string)
search       (string)
```

```
   window.location = „http://www.dannyg.com“
```

Methods

```
(None)
```

Event Handlers

```
(None)
```

History Object

Property

```
length     (integer)
  histLength = history.length
```

Methods

```
back()
forward()
go(relativeNumber | „URLOrTitleSubstring“)
  // refresh current frame
  history.go(0)
```

Event Handlers

```
 (None)
```

Document Object

Creating a Document

```
<BODY
     [BACKGROUND="backgroundImageURL"]
     [BGCOLOR="#backgroundColor"]
     [TEXT="#foregroundColor"]
     [LINK="#unfollowedLinkColor"]
     [ALINK="#activatedLinkColor"]
     [VLINK="#followedLinkColor"]
     [onLoad="handlerTextOrFunction"]
     [onUnload="handlerTextOrFunction"]>
</BODY>
```

Properties

```
forms(array)
location     (string)
title(string)
alinkColor   (hexadecimal triplet or constant)
vlinkColor   (hexadecimal triplet or constant)
bgColor      (hexadecimal triplet or constant)
fgColor      (hexadecimal triplet or constant)
linkColor    (hexadecimal triplet or constant)
lastModified (date string)
anchors      (array)
links(array)
referrer     (string)
cookie       (string)
   docTitle = document.title
```

Methods

```
write(„string“)
writeln(„string“)
open([„mimeType“])
close()
clear()
   document.writeln(„<H1>Howdy</H1>“)
```

Event Handlers

```
 (None)
```

Form Object

Creating a Form Object

```
<FORM
     NAME="formName"
     [TARGET="windowName"]
     [ACTION="serverURL"]
     [METHOD=GET | POST]
     [ENCTYPE="MIMEType"]
     [onSubmit="handlerTextOrFunction"] >
</FORM>
```

Properties

```
elements     (array)
action       (URL)
method       (GET or POST)
target       (window name)
encoding     (MIME type)
   howMany = document.forms[0].elements.length
```

Method

```
submit()
  document.forms[0].submit()
```

Event Handler

```
onSubmit=
```

Text, Textarea, & Password Objects

Creating a Text Object

```
<FORM>
<INPUT
     TYPE="text"
     NAME="fieldName"
     [VALUE="contents"]
     [SIZE="characterCount"]
     [onBlur="handlerTextOrFunction"]
     [onChange="handlerTextOrFunction"]
     [onFocus="handlerTextOrFunction"]
     [onSelect="handlerTextOrFunction"]>
</FORM>
```

Creating a Textarea Object

```
<FORM>
<TEXTAREA
     NAME="fieldName"
     ROWS="rowCount"
     COLS="columnCount"
     [onBlur="handlerTextOrFunction"]
     [onChange="handlerTextOrFunction"]
     [onFocus="handlerTextOrFunction"]
     [onSelect="handlerTextOrFunction"]>
     defaultText
</TEXTAREA>
</FORM>
```

Creating a Password Object

```
<FORM>
<INPUT
     TYPE="password"
     NAME="fieldName"
     [VALUE="contents"]
     [SIZE="characterCount"]>
</FORM>
```

Properties

```
value(string)*
name (string)
defaultValue (string)
*For password object, only the default value
specified in a VALUE= attribute.
  var cityString = document.forms[0].city.value
```

Methods

```
select()
focus()
blur()
```

Event Handlers

```
onChange=
onFocus=
onBlur=
onSelect=
```

Hidden Object

```
Creating a Hidden Object
<FORM>
<INPUT
     TYPE="hidden"
     NAME="fieldName"
     [VALUE="contents"]>
</FORM>
```

Properties

```
value(string)
name (string)
defaultValue (string)
```

Methods

```
 (None)
```

Event Handlers

```
 (None)
```

Button, Submit, & Reset Objects

Creating Button Objects

```
<FORM>
<INPUT
     TYPE="text" | „submit“ | „reset“
     NAME="buttonName"
     VALUE="buttonLabelString"
     [onClick="handlerTextOrFunction"] >
</FORM>
```

Properties

```
value(string)
name (string)
  document.forms[0].clickMe.value
```

Method

```
click()
```

Event Handler

```
onClick=
```

Checkbox Object

Creating a Checkbox Object

```
<FORM>
<INPUT
     TYPE="checkbox"
     NAME="boxName"
     VALUE="buttonValue"
     [CHECKED]
     [onClick="handlerTextOrFunction"]>
     buttonText
</FORM>
```

Properties

```
checked      (Boolean)
name         (string)
value        (string)
defaultChecked      (Boolean)
   var setting = document.forms[0].citizen.checked
```

Method

```
click()
```

Event Handler

```
onClick=
```

Radio Object

Creating a Radio Object

```
<FORM>
<INPUT
     TYPE="radio"
     NAME="buttonGroupName"
     VALUE="buttonValue"
     [CHECKED]
     [onClick="handlerTextOrFunction"]>
     buttonText
</FORM>
(All buttons in a group must have the same NAME
assigned to them.)
```

Properties

```
checked      (Boolean)
name         (string)
length              (integer)
value        (string)
defaultChecked      (Boolean)
  var isModern = document.forms[0].style[2].checked
```

Method

```
onClick()
Event Handler
onClick=
```

Select Object

Creating a Select Object

```
<FORM>
<SELECT
     NAME="listName"
     [SIZE="number"]
     [MULTIPLE]
     [onBlur="handlerTextOrFunction"]
     [onChange="handlerTextOrFunction"]
     [onFocus="handlerTextOrFunction"]>
     <OPTION [SELECTED] [VALUE="string"]>listItem
     [...<OPTION [VALUE="string"]>listItem]
</SELECT>
</FORM>
```

Properties

```
selectedIndex               (integer)
length              (integer)
name         (string)
options[index]              (array)
options[index].text (string)
options[index].value(string)
options[index].selected     (Boolean)
options[index].index(integer)
options[index].defaultSelected     (Boolean)
  var choice = document.forms[0].popup1.
     options[document.forms[0].popup1.
     selectedIndex].value
```

Methods

```
 (None)
```

Event Handler

```
onChange=
```

Link Object

Creating a Link Object

```
<A HREF="locationOrURL"
     [NAME="anchorName"]
     [TARGET="windowName"]
     [onClick="handlerTextOrFunction"]
     [onMouseOver="handlerTextOrFunction"]>
     linkDisplayTextOrImage
</A>
```

Properties

```
links[index].target (window name)
length              (integer)
   var linkCount = document.links.length
```

Methods

(None)

Event Handlers

```
onMouseOver=         (must end with ;return true)
onClick=
```

Anchor Object

Creating an Anchor Object

```
<A NAME="anchorName">
     anchorDisplayTextOrImage
</A>
```

Properties

(None)

Methods

(None)

Event Handlers

(None)

Navigator Object

Properties

```
appName      (string)
appVersion   (string)
appCodeName  (string)
userAgent    (string)
```

Methods

(None)

Event Handlers

(None)

String Object

Property

```
string.length
    var strLen = „Howdy Doody“.length
```

Methods

```
string .toLowerCase()
string .toUpperCase()
string .indexOf(searchString  [, startIndex])
string .lastIndexOf(searchString  [, startIndex])
string .charAt(index)
string .substring(indexA , indexB)
string .anchor(„anchorName“)
string .big()
string .blink()
string .bold()
string .fixed()
string .fontcolor(colorValue)
string .fontsize(integer1to7)
string .italics()
string .link(locationOrURL)
string .small()
string .strike()
string .sub()
string .sup()
```

Event Handlers

(None)

Math Object

Properties

```
Math.E
Math.LN2
Math.LN10
Math.LOG2E
Math.LOG10E
Math.PI
Math.SQRT1_2
Math.SQRT2
  var circumf = Math.PI * 4
```

Methods

```
Math.abs(val)
Math.acos(val)
Math.asin(val)
Math.atan(val)
Math.ceil(val)
Math.cos(val)
Math.exp(val)
Math.floor(val)
Math.log(val)
Math.max(val1, val2)
Math.min(val1, val2)
Math.pow(val1, val2)
Math.random()*
Math.round(val)
Math.sin(val)
Math.sqrt(val)
Math.tan(val)
*Not available in Navigator 2.0 for Windows or Mac.
  var circArea = Math.PI * Math.pow(diam,2)
```

Event Handlers

(None)

Date Object

Properties

(None)

Methods

```
dateObj .getTime()   (0-...)
dateObj .getYear()   (70-...)
dateObj .getMonth()  (0-11)
dateObj .getDate()   (1-31)
dateObj .getDay()    (0-6)
dateObj .getHours()  (0-23)
dateObj .getMinutes()       (0-59)
dateObj .getSeconds()       (0-59)
dateObj .setTime(val)       (0-...)
dateObj .setYear(val)       (70-...)
dateObj .setMonth(val)      (0-11)
dateObj .setDate(val)       (1-31)
dateObj .setDay(val) (0-6)
dateObj .setHours(val)      (0-23)
dateObj .setMinutes(val)    (0-59)
dateObj .setSeconds(val)    (0-59)
```

```
dateObj .getTimezoneOffset()       (0-...)
dateObj .toGMTString()      (string)
dateObj .toLocaleString()   (string)
Date.parse(„dateString“)
Date.UTC(date value )
In Navigator 2.0 for Macintosh, most date methods
produce erroneous results.
```

Event Handlers

(None)

Programmablauf

if decisions

```
if (condition)
     statementsIfTrue
if (condition)
     statementsIfTrue
 else
     statementsIfFalse
```

for loops

```
for ([initial expr] ; [condition] ; [update expr]) {
     statements
}
    for (var i = 0; i <= maxValue; i++){
  }

for (var  in object ) {
     statements
}
```

while loop

```
while (condition) {
     statements
}
```

with statement

```
with (object) {
     statements
}
   with (Math) {
     var circArea = PI * pow(diam,2)
   }
```

Conditional Expressions

```
variable  = (condition) ? val1  : val2
   var sign = (if x >= 0) ? „positive“ : „negative“
```

Comparison Operators

```
==   Equals
!=   Does not equal
>    Is greater than
>=   Is greater than or equal to
<    Is less than
<=   Is less than or equal to
```

Connubial Operators

```
+    Plus and string concatenate
-    Minus
*    Multiply
/    Divide
%    Modulo
++   Increment
–    Decrement
-val Negation
```

Assignment Operators

```
=    Equals
+=   Add by value
-=   Subtract by value
*=   Multiply by value
/=   Divide by value
%=   Modulo by value
```

Boolean Operators

```
&&   And
||   Or
!    Not
```

Bitwise Operators

```
&    Bitwise And
|    Bitwise Or
^    Bitwise XOR
<<   Left Shift
>>   Right Shift
>>>  Zero Fill Right Shift
```

windowObject = window.open([parameters])

Warum sind Datenbanken etwas unnatürliches ???
Alle normalen Worter schreibt man im Zehnfinger-System mit beiden Händen, aber DATABASE schreibt man nur mit der linken Hand.