

Dynamische Webseitengestaltung

Dynamisches HTML (engl. "Dynamic HTML" oder abgekürzt "**DHTML**") ist eine Erfindung von Marktstrategen, sagen Kritiker. In der Tat ist Dynamisches HTML keine klassische HTML-Erweiterung in Gestalt neuer HTML-Elemente. Es ist auch keine neue Sprache. Dynamisches HTML ist vielmehr der Sammelbegriff für verschiedene Lösungen, um dem Autor einer Web-Seite zu ermöglichen, Elemente der Webseite während der Anzeige dynamisch zu ändern, sei es automatisch oder durch Einwirken des Anwenders.

Cascading Stylesheets (CSS) sind eine unmittelbare Ergänzung zu HTML. Es handelt sich dabei um eine Sprache zur Definition von Formateigenschaften einzelner HTML-Elemente. Sie können CSS für ein einzelnes HTML-Tag anwenden, im Kopf einer HTML-Datei zentrale Definitionen zum Aussehen aller Tags einer Art oder Sie können Ihre Stylesheet-Definitionen sogar in separaten Dateien notieren. Die Stylesheet-Dateien können Sie in beliebig vielen HTML-Dateien referenzieren. Auf diese Weise können Sie für große Projekte einheitliche Layouts entwerfen. Mit ein paar kleinen Änderungen in einer zentralen Stylesheet-Datei können Sie dann für Hunderte von HTML-Dateien ein anderes Aussehen bewirken.

JavaScript ist kein direkter Bestandteil von HTML sondern eine eigene Programmiersprache. Diese Sprache wurde jedoch eigens zu dem Zweck geschaffen, HTML-Autoren ein Werkzeug in die Hand zu geben, mit dessen Hilfe sich Webseiten optimieren lassen. JavaScripts werden wahlweise direkt in der HTML-Datei oder in separaten Dateien notiert. Sie werden zur Laufzeit vom Webbrowser interpretiert. Dazu besitzen moderne Webbrowser entsprechende Interpreter-Software. JavaScript wurde 1995 von Netscape eingeführt und lizenziert. Netscape diktierte also, woraus das "offizielle" JavaScript bestand. Mittlerweile ist die Situation aus verschiedenen Gründen unübersichtlicher geworden...

Das **Document Object Model (DOM)** ist eine vom W3-Konsortium verabschiedete Norm, die zunächst einmal den Zugriff einer Programmiersprache auf beliebige Elemente eines Auszeichnungssprachen-Dokuments beschreibt. Das DOM ist also weder selber eine eigene Programmiersprache, noch ist es auf HTML beschränkt. Es definiert lediglich Objekte, Eigenschaften und Methoden, die eine Programmiersprache umsetzen sollte, wenn sie sich als DOM-fähig bezeichnen will. Anwendbar sollen diese Objekte, Eigenschaften und Methoden auf alle Dokumente sein, die in einer XML-gerechten Auszeichnungssprache geschrieben sind. JavaScript entspricht dieser DOM-Norm.

Java ist eine 1995 von der Firma Sun entwickelte, Plattform (Betriebssystem) unabhängige Programmiersprache und wird daher häufig auch in Webseiten eingesetzt, um animierte oder interaktive Inhalte (wie z.B. Spiele) zu erzeugen. Java-Programme, die zur Anzeige innerhalb des Browserfensters erstellt wurden, werden als Java-Applets bezeichnet. Die Programmierung mit Java ist komplex und sprengt den Rahmen dieses Kurses. In kompilierter Form mit der Erweiterung *.class werden sie in die HTML-Datei durch das Tag <object> (früher auch <applet>) eingebunden. Das Entwicklerkitt ist für private Zwecke kostenfrei.

Das von der Firma Macromedia entwickelte **Flash** bietet die Möglichkeit, Multimedia-Effekte auf Web-Seiten zu bringen, aber auch Anwendungen wie Spiele, Simulationen oder Navigations-Unterstützung für Web-Seiten. Es gilt momentan als sehr modern und ist der neueste Vertreter der Gattung "ich mach HTML tot" – zumindest ist es von jenen so hochstilisiert worden, die sich mit HTML und den offenen Standards noch nie abfinden konnten. Da hinter Flash ein kommerzielles Software-Produkt steckt, das in keiner Weise mit den offenen Internet-Standards vergleichbar ist, werden also Äpfel mit Birnen verglichen. Flash steht vor allem dort hoch im Kurs, wo man der Meinung ist, auf einer Web-Seite müsse es zugehen wie im Fernsehen. Manche Kreativlinge unterliegen diesem Glauben, aber noch mehr die Marketing-Abteilungen von Firmen, die bei Agenturen ihren Web-Auftritt gestalten lassen und dafür am Bildschirm etwas sehen wollen, von dem sie glauben, das sei besonders schwierig und atemberaubend. Alles in allem ist Flash also ein mächtiges (und kostenpflichtiges) Werkzeug, um Inhalte zu visualisieren. Auch Flash muss in eine erste HTML-Seite eingebettet werden.

Nicht unerwähnt bleiben sollte jedoch, dass es auch frei verfügbare, offen dokumentierte und letztlich noch leistungsfähigere Internet-Standards für das gibt, was Flash leistet. So werden beim W3-Konsortium zwei **XML**-basierte Sprachen namens **SVG** und **SMIL** standardisiert, die in Verbindung mit JavaScript und DOM ähnliche Möglichkeiten eröffnen.

Literatur-Tipps

Lehrgänge zum Selbststudium

HTML 4 für Dummies

Ed Tittel & Natanya Pitts
mitp-Verlag
ca. 25,00 €

JavaScript für Dummies

Emily A. Vander Veer
mitp-Verlag
ca. 25,00 €

Web-Academy (X)HTML

Mechthild Käufer
Data Becker
ca. 25,00 €

Web-Academy JavaScript

Antje Hofmann
Data Becker
ca. 25,00 €

Kurzreferenzen (Taschenbücher, Online)

HTML – kurz & gut

Jennifer Niederst

CSS – kurz & gut

Eric A. Meyer

JavaScript – kurz & gut

David Flanagan

Alle drei bei: O'Reilly Verlag
jeweils ca. 10,00 €

SelfHTML Version 8.1

Stefan Münz

TeamOne, München

Kostenlos unter <http://de.selfhtml.org/>

Script-Bibliotheken (Online)

Andre's JavaScripte

Andre Göntgen

<http://www.javascripte.org/>

brauchbar.de

Thomas Salvador

<http://www.brauchbar.de/wd/artikel/list-js.html>

HTML-World

Jan Winkler

<http://www.html-world.de/>

JRP Webdesign

Johann Piber

<http://www.jrp.at/beispiele/>

jswelt.de

Michael Ohnesorg & Tobias Bohlinger

<http://www.jswelt.de/>

Michael Mailer's JavaScripte

Michael Mailer

<http://www.mywebaid.de/>

PC-Anfänger

Antje Hofmann

<http://pc-anfaenger.de/>

Allgemeine Informationen zum Computer und zum Internet

Duden – Basiswissen Schule

Angewandte Informatik

Dr. Lutz Engelmann (Hrsg.)

Dudenverlag

21,00 €

Taschenbuch Multimedia

Peter A. Henning

Fachbuchverlag Leipzig

ca. 20,00 €

Cascading Stylesheets (CSS)

Genau wie HTML wird auch CSS vom W3-Konsortium normiert. Es handelt sich also um einen Firmen unabhängigen, offen dokumentierten und frei verwendbaren Standard. Für CSS gibt es genau wie für HTML eine Arbeitsgruppe beim W3-Konsortium, die sich um die Weiterentwicklung der Sprache kümmert und sich dabei dem Regelwerk zur Entstehung der so genannten Recommendations (Empfehlungen) des W3-Konsortiums unterzieht.

Wie bei HTML, so gibt es auch bei CSS Sprachversionen, die auf den Neuauflagen der Recommendations basieren. Die Version 1.0 von CSS wurde bereits im Jahr 1996 herausgegeben und 1999 nochmals überarbeitet. 1998 erhielt die Version 2.0 den Status einer Empfehlung. Mittlerweile wird an Version 3.0 gearbeitet.

Die Einbettung von CSS in ein HTML-Dokument

<pre><head> ... <style type="text/css" Selektor {Attribut: Wert;} ... /> ... </head></pre>	<pre><head> ... <style type="text/css"> <!-- Selektor {Attribut: Wert;} --> ... </style> ... </head></pre>
als Attribute des Style-Tags	als Inhalt des Style-Tags, umschlossen von Kommentarzeichen
<pre><head> ... <link rel="stylesheet" type="text/css" href="url"> ... </head></pre>	<pre><body> ... <p style="Attribut: Wert;..."> ... </p> ... </body></pre>
als Link auf ein externes CSS-Dokument	als Spezifikation in das entsprechende HTML-Tag

Syntax und Selektoren

Jede Anweisung besteht aus einem Selektor, der sich auf ein HTML-Tag bezieht, und einer Deklaration in geschwungenen Klammern. Die Deklaration besteht aus mindestens einem Attribut und nach einem Doppelpunkt dem zugewiesenen Wert. Die Attribute werden mit Semikolon voneinander getrennt.

```
h1 {color: red; text-align: center; } /* Kommentar */
```

Mehreren Selektoren kann des gleiche Format zugewiesen werden.

```
h1, h2, p {color: green; }
```

Selektoren können kontextsensitiv verknüpft werden. Im folgenden Beispiel wird nur der gefettete Bereich des Absatzes blau dargestellt.

```
p b {color: blue; }
```

Durch die HTML-Spezifikation class können verschiedene Selektoren zusammengefasst oder ein bestimmter Selektor isoliert werden.

```
In HTML: <h1>Warnung</h1><h1 class="achtung">Gefahr!</h1>
          <p class="achtung">Vorsicht vor dem Hund!</p>
```

```
In CSS:  h1 {text-align: center; } für alle h1-Tags
          h1.achtung {color: red; } für das h1-Tag der Klasse „achtung“
          .achtung {background: yellow; } für alle Tags der Klasse „achtung“
```

Pseudo-Klassen-Selektoren

Speziell für die Darstellung der unterschiedlichen Zustände von Links (in HTML mit `` erzeugt) stellt CSS einige Pseudo-Klassen bereit. D.h. sie müssen im HTML-Tag nicht benannt werden.

```
a:link {color: blue; }  
a:active {color: red; }  
a:visited {color: purple; }
```

Auch für andere Elemente gibt es Pseudoklassen.

```
p:first-letter {color: red} schreibt den ersten Buchstaben des Absatzes rot  
p:first-line {color: green} schreibt die erste Zeile des Absatzes grün
```

Werte und ihre Definitionen

Den Attributen der Elemente müssen Werte zugewiesen werden. Dies können Schlüsselwörter sein (wie `red`, `bold`, `center`), Maßangaben, Prozentwerte, Hexadezimalcode oder URLs.

Größen können absolut in Zentimeter, Millimeter, Inch aber auch in Punkt und Pica und in Pixel angegeben werden. Die Standardschrift eines Geschäftsbriefs liegt bei 10–12 Punkt. Die Angaben in cm, mm, in werden nur sehr ungenau umgesetzt. Pixel werden von den meisten Browsern als Bildschirmpixel interpretiert (z.B. 1027*768 Auflösung).

```
10mm = 1cm = 0,394in  
1in = 2,54cm = 25,4 mm  
  
12pt (Punkt) = 1 pc (Pica) [= 1/6 Inch]  
72pt = 6pc [=1 Inch]
```

Sie können die Größe eines Elements auch relativ angeben. Dafür stehen Ihnen Prozentangaben zur Verfügung oder die Einheit `em`. Prozent können positiv und negativ vorkommen und im Betrag auch deutlich über 100 hinausgehen. Sie verhalten sich immer relativ zu anderen Werten.

```
{line-height: 150%} setzt den Zeilenabstand auf das anderthalbfache der Zeilenhöhe  
{line-height: 1.5em} erzielt dasselbe Ergebnis
```

Farbangaben können Sie mit Schlüsselwörtern (die 16 VGA-Farben auf Englisch) machen, mit Prozenten (0–100) oder in Hexadezimalcode (0–F oder 0–255). Die beiden letzten Methoden arbeiten mit den RGB-Kanälen. Bei Prozentangaben sind Dezimalwerte erlaubt (z.B. 75.5%).

```
#FF0000 = rgb(255,0,0) = rgb(100%,0%,0%) = red  
#0000FF = rgb(0,0,255) = rgb(0%,0%,100%) = blue
```

URLs finden selten in CSS Verwendung. Wenn Sie einer Datei ein Hintergrundbild zuweisen, denken Sie an die Lesbarkeit des Textes und an die Dateigröße. Relative Links werden meist relativ zur CSS-Datei interpretiert. Einige Browser lesen sie aber relativ zur HTML-Datei. Daher sind absolute Links sicherer.

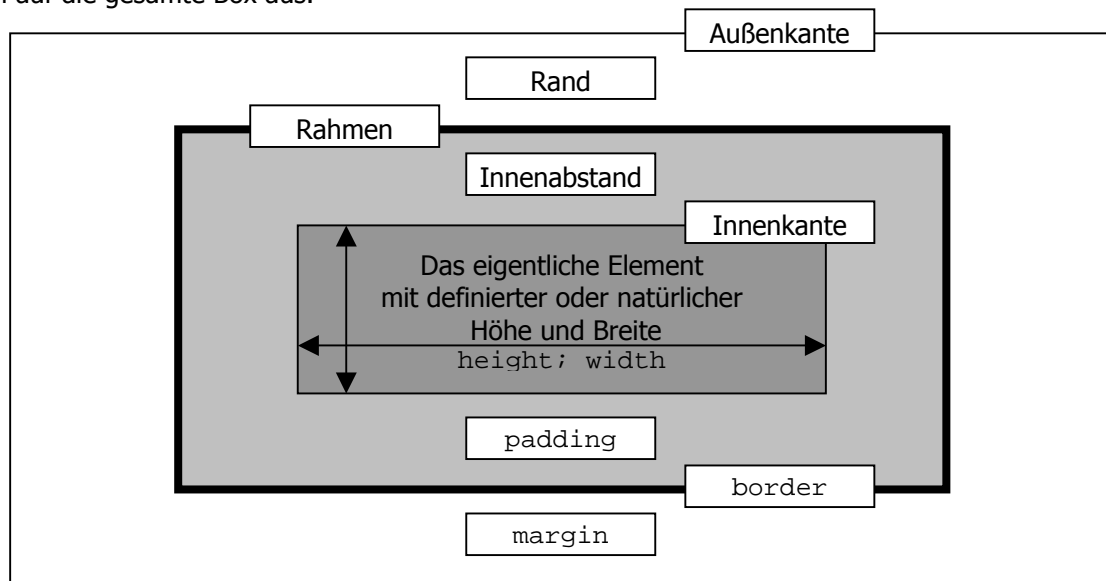
```
body {background-image: url(bild.jpg); }  
body {background-image: url(http://www.meineseite.de/bild.jpg); }
```

Elementklassifizierung

- **Block-Elemente**
Absätze (`p`), Überschriften (`h1–h6`), Listen (`ul`, `ol`), Tabellen (`table`), (`div`) und (`body`)
Sie beginnen in einer neuen Zeile und dulden kein Folge-Element in dieser Zeile.
- **Inline-Elemente**
Anker (`a`), Text hervorhebungen (`em`), (`span`) sowie die meisten Grafiken (`img`)
Sie erzeugen keinen Zeilenumbruch vor oder nach dem Element.
- **Listenpunkt-Elemente**
Die Punkte in einer Aufzählung (`li`) dienen der Zuweisung eines Gliederungszeichens oder der Durchnummerierung

Box- und Rahmeneigenschaften

Alle Elemente erzeugen um sich ein virtuelles Rechteck (Box). Die Formatierung eines Elements wirkt sich auf die gesamte Box aus.



- Der Hintergrund eines Elements (Grafik oder Farbe) erstreckt sich über Inhaltsbereich und Innenabstand bis zum Rahmen.
- Die Attribute `padding`, `border` und `margin` können so für das gesamte Element oder speziell für den rechten (`border-right`), linken (`-left`), oberen (`-top`) oder unteren (`-bottom`) Bereich bestimmt werden.
- Höhe (`height`), Breite (`width`) und der Rand (`margin`) dürfen auf den Wert `auto` gesetzt werden. Die Ränder dürfen zudem noch negative Werte annehmen.
- Die Innenabstände (`padding`) und der Rahmen (`border`) können keine negativen Werte annehmen. Sie sind standardmäßig auf 0 gesetzt.

Referenz der CSS1 Attribute (Auswahl)

<code>background-color: FARBE;</code>	benennt eine Hintergrundfarbe Standardwert: transparent
<code>background-image: URL;</code>	benennt ein Hintergrundbild Standardwert: none
<code>background-repeat: WORT</code>	Wiederholungsart des Hintergrundbildes Werte: <code>repeat repeat-x repeat-y no-repeat</code> Standardwert: <code>repeat</code>
<code>border: STYLE WIDTH COLOR</code>	definiert den Element-Rahmen Werte für STYLE: <code>none dotted dashed solid double groove ridge inset outset</code> Standardwerte: <code>none - Farbe des Elements</code>
<code>color: FARBE</code>	Vordergrundfarbe des Elements Standardwert: Browserabhängig
<code>font-family: SCHRIFTNAME</code>	definiert eine Schriftart oder -familie z.B.: <code>Arial, Garamond, sans-serif, monospace</code> Standardwert: Browserabhängig

<code>font-size: WORT % ZAHL</code>	setzt die Schriftgröße in % oder pt, pc, in, cm oder: xx-small ... medium ... xx-large oder: smaller larger Standardwert: medium
<code>font-style: WORT</code>	definiert eine Schrifteigenschaft italic oblique normal Standardwert: normal
<code>font-weight: WORT ZAHL</code>	definiert die Stärke der Schrift 100–900 normal bold bolder lighter Standardwert: 400 normal
<code>height: ZAHL auto</code>	definiert die Höhe eines Objekts Standardwert: auto
<code>list-style: TYPE POSITION IMAGE</code>	definiert Listen und Aufzählungen Werte für TYPE: disc circle square decimal lower-/upper-roman lower-/upper-alpha none Werte für POSITION: inside outside Werte für IMAGE: url(*.gif) Standardwerte: disc outside
<code>margin: % ZAHL {1,4}</code>	definiert den Rand Standardwert: 0
<code>padding: % ZAHL {1,4}</code>	definiert den Innenabstand Standardwert: 0
<code>text-align: WORT</code>	richtet Blockelemente aus Werte: left right center justify Standardwert: left
<code>text-decoration: WORT</code>	benennt einige Texteffekte Werte: none under/overline line-through blink Es werden nicht immer alle unterstützt Standardwert: none Standardwert <a>: underline
<code>text-indent: ZAHL</code>	definiert eine Einrückung der ersten Zeile Standardwert: 0
<code>text-transform: WORT</code>	steuert die Groß- und Kleinschreibung Werte: capitalize upper/lowercase none Standardwert: none
<code>vertical-align: % WORT</code>	definiert die vertikale Ausrichtung der Grundlinie eines Elements Werte: sub (text-)top middle (text-)bottom Standardwert: baseline
<code>width: ZAHL auto</code>	definiert die Breite des Elements Standardwert: auto

Ein Beispiel für CSS2 in HTML

```
<div style="position:absolute; top:100px; left:100px;
    clip:rect(0px 130px 130px 0px);">

</div>
```

JavaScript

Bevor Sie JavaScript-Code programmieren, müssen Sie sich exakt darüber im klaren sein, welches Problem Sie damit lösen wollen, was man mit HTML selbst alles machen kann und wo die Grenzen von HTML liegen. Von JavaScript müssen Sie dann so viel wissen, dass Sie entscheiden können, ob das Problem mit JavaScript überhaupt lösbar ist. JavaScripts werden im Browser des Anwenders ausgeführt, nicht auf dem Server, wo die Webseiten abgelegt sind. Das heißt, JavaScript kann vom Browser erst dann ausgeführt werden, wenn er eine Webseite gerade einliest oder nachdem er sie eingelesen hat – dann auch ausgelöst durch Benutzerereignisse wie z.B. Mausklicks.

Aus diesem Grund können Sie mit JavaScript also keine Anwendungen wie Gästebücher oder Web-Foren programmieren. Denn solche Anwendungen müssen die Daten aller beitragenden Anwender zentral auf dem Server speichern. Wohl aber kann JavaScript unterstützend eingesetzt werden. So ist es beispielsweise sinnvoller, Formulareingaben des Anwenders vor dem Absenden mit Hilfe von JavaScript zu überprüfen, als den Server-Rechner mit dieser Arbeit zusätzlich zu belasten.

Es lohnt sich durchaus, im Netz nach frei verfügbaren JavaScript-Beispielen zu suchen, die genau Ihr Problem lösen. Denn auf Code zurückzugreifen, der sich im Einsatz bereits bewährt hat, erspart Ihnen viel Ausprobieren. In vielen Fällen genügt es, vorhandene JavaScripts den eigenen Erfordernissen anzupassen. Bei vorhandenen JavaScripts müssen Sie allerdings so viel von der Sprache verstehen, dass Sie genau wissen, welche Variablen oder festen Werte Sie verändern oder welche Funktion Sie ergänzen wollen.

Die Einbettung von JavaScript in ein HTML-Dokument

<pre><head> ... <script type="text/javascript"> <!-- JavaScript Anweisungen ... //--> </script> ... </head></pre>	<pre><head> ... <script type="text/javascript" src="url"> </script> ... </head></pre>
als Inhalt des Style-Tags, umschlossen von Kommentarzeichen	als Link auf ein externes JS-Dokument

Grundlagen der JavaScript-Syntax

JavaScript ist objektorientiert. Objekte sind die Einheiten, denen Eigenschaften, Methoden und Unterobjekte zugeordnet sind. Methoden sind Funktionen, die Aktionen ausführen, aber im Gegensatz zu alleinstehenden Funktionen an ein bestimmtes Objekt gebunden sind. Viele vordefinierte JavaScript-Objekte haben Methoden.

JavaScript besteht letztendlich aus einer kontrollierten Anordnung von Anweisungen. Das sind Befehle, die der JavaScript-Interpreter des Browsers bewertet und in Maschinencode umsetzt, der auf dem Rechner des Anwenders ausführbar ist. Eine Anweisung in JavaScript besteht immer aus einem Befehl, der mit einem Semikolon [;] oder einem Zeilenumbruch abgeschlossen wird. Zur Sicherheit sollten Sie sich beides angewöhnen. Ein Anweisungsblock wird durch eine geschweifte Klammer { begonnen und durch eine geschweifte Klammer } beendet.

Eine Anweisung ist zum Beispiel:

- wenn Sie einer Variablen einen Wert zuweisen [Zahl = 42;]
- wenn Sie eine Operation durchführen [Ergebnis = Zahl * Zahl;]
- wenn Sie einen bedingten Befehl ausführen [if(Zahl > 1000) Zahl = 0;]
- wenn Sie eine Funktion oder eine Methode aufrufen, bzw. wenn Sie eine Objekteigenschaft auslesen oder ändern [alert("Das Quadrat von " + Zahl + " = " + Ergebnis);]

Wenn Sie Ihren JavaScript-Code kommentieren wollen, um bei einer späteren Bearbeitung gleich zu wissen, wofür Sie das Script geschrieben haben, können Sie dies im Script selbst tun:

```
// Ein einzeiliger Kommentar.  
/* Ein mehrzeiliger Kommentar */
```

Achten Sie auf Groß- und Kleinschreibung. JavaScript ist casesensitive. Manche Wörter oder einzelne Zeichen sind bei JavaScript verboten, da sie mit spezifischen Bedeutungen belegt sind. Strings werden durch doppelte oder einfache Anführungsstriche gekennzeichnet. Wenn nun im String Anführungsstriche benutzt werden sollen, müssen sie zuerst durch ein Backslash entwertet oder durch einfache/doppelte ersetzt werden.

```
"String mit `Wort` in Anführungszeichen"  
"String mit \"Wort\" in Anführungszeichen"  
  
\\ = ein einfacher Backslash als String  
\n = ein Zeilenumbruch im String
```

Das Objekt `window` (Fenster) ist das oberste Objekt der Objektfamilie für alles, was im Browser-Fenster angezeigt wird. Das aktuelle Fenster des Browsers können Sie über die Objekte `window` oder `self` ansprechen. Mit der folgenden Anweisung wird die Anzeige in der Statusleiste bestimmt.

```
window.defaultStatus = "Meine Homepage";
```

Die nächste Anweisung fragt nach dem Namen des Objekts `navigator` und gibt diesen in Verbindung mit einem String in die geladene HTML-Datei (das Objekt `document`) aus.

```
document.write("So, so, Sie verwenden also " + navigator.appName);
```

Die erste Funktion

Nun folgt schon die erste selbst geschriebene Funktion mit dem Namen `NeuFenster`. Es wird die Variable `MeinFenster` definiert, die aus dem Objekt `window` mit der Methode `open` und einigen Eigenschaften besteht. Schließlich wird das Objekt `MeinFenster` mit der Methode `focus` in den Vordergrund gestellt oder aktiviert.

```
function NeuFenster() {  
    MeinFenster = window.open("datei2.htm", "Zweitfenster",  
    "width=300,height=200,scrollbars");  
    MeinFenster.focus();  
}
```

Wenn nun im Body der HTML-Datei folgender Link eingesetzt wird, öffnet dieser beim Klick ein neues Fenster mit vorbestimmten Eigenschaften.

```
<a href="javascript:NeuFenster()">Hier klicken für neues Fenster</a>
```

Mit diesem Link kann das Hauptfenster das neu geöffnete Fenster auch wieder schließen.

```
<a href="javascript:MeinFenster.close()">Fenster schließen</a>
```

Im HTML-Dokument des zweiten Fensters könnte der nächste Link untergebracht sein, der das Hauptfenster, den `opener`, schließt.

```
<a href="javascript:opener.close()">Hauptfenster schließen </a>
```

Mit dieser zweiten Funktion lässt sich nun prüfen, ob das Zweitfenster geschlossen wurde. Je nach dem wird ein Dialogfenster mit einem Hinweis ausgegeben.

```
function CheckOpen() {  
    if(MeinFenster.closed == true) alert("Fenster wurde geschlossen");  
    else alert("Fenster noch offen");  
}
```


Anweisungen

Von den JavaScript-Anweisungen möchte ich hier sechs kurz vorstellen. Einige haben wir in den Beispielen weiter oben schon verwendet. Lassen Sie sich bitte nicht von der doppelten Verwendung des Begriffs Anweisung irritieren.

<code>function</code>	Sie definiert eine Funktion, z.B. diese namens Funktionsname, deren Body aus einer oder mehr Anweisungen besteht und die mit keiner oder mehr mit Komma voneinander getrennten Argumenten versehen ist. <pre>function Funktionsname (Argumente) { Anweisungen; } }</pre>
<code>if/else</code>	Sie führt eine Anweisung aus, wenn ein Ausdruck wahr ist, sonst führt sie eine andere Anweisung aus. <pre>if (Ausdruck) {Anweisungen1; } [else {Anweisungen2; }]</pre>
<code>return</code>	Sie beendet die Ausführung der laufenden Funktion und kehrt zu ihrem Aufrufer zurück. Wenn ein Ausdruck folgt, wird dessen Wert als Rückgabe der Funktion verwendet. <pre>return [Ausdruck] ;</pre>
<code>switch/case</code>	Sie prüft, ob ein berechneter Ausdruck einem bestimmten konstanten Wert entspricht und führt die zugeordneten Anweisungen aus. Gibt es keine Entsprechung, werden die default Anweisungen ausgeführt. <pre>switch (Ausdruck) { case konstanterWert1: Anweisungen1; [case konstanterWert2: Anweisungen2;] default: Anweisungen; } }</pre>
<code>var</code>	Sie deklariert und initialisiert (optional) eine oder mehr Variablen. <pre>var Name [= Wert] [, Name2 = Wert2]</pre>
<code>while</code>	Sie ist eine elementare Schleife. Sie führt eine Anweisung aus, so lange ein Ausdruck wahr ist. <pre>while (Ausdruck) {Anweisung; } }</pre>

Objekte und ihre Methoden

Die Methoden sind Aktionen, die ein bestimmtes Objekt ausführen kann. Die Auswahl möglicher Methoden ist demnach abhängig vom jeweiligen Objekt. In der Folge stelle ich Ihnen nochmals drei Objekte und eine Auswahl der möglichen Methoden vor

Mit dem Objekt `window` haben wir weiter oben schon gearbeitet. Lassen Sie mich an dieser Stelle einige Methoden in alphabetischer Reihenfolge benennen:

<code>alert()</code>	Ruft ein Dialogfenster mit Informationen auf.
<code>blur()</code>	Verlässt das Fenster und setzt es ganz nach hinten.
<code>close()</code>	Schließt das Fenster.
<code>confirm()</code>	Ruft ein Dialogfenster zur Bestätigung auf.
<code>focus()</code>	Aktiviert das Fenster und setzt es somit nach vorn.
<code>open()</code>	Öffnet ein neues Fenster.
<code>print()</code>	Druckt den Fensterinhalt aus.
<code>prompt()</code>	Ruft ein Dialogfenster zur Werteingabe auf.

Event-Handler

Event-Handler (Ereignis-Behandler) sind ein wichtiges Bindeglied zwischen HTML und JavaScript. Event-Handler werden meist in Form von Attributen in HTML-Tags notiert. Da es sich um Bestandteile handelt, die innerhalb von HTML vorkommen, hat das W3-Konsortium die Event-Handler mittlerweile auch in den HTML-Sprachstandard mit aufgenommen.

Event-Handler erkennen Sie daran, dass solche HTML-Attribute immer mit `on` beginnen, zum Beispiel `onClick=`. Hinter dem Gleichheitszeichen notieren Sie in Anführungszeichen, eine JavaScript-Anweisung. Wenn Sie mehrere Anweisungen ausführen wollen, dann definieren Sie sich dazu am besten in einem JavaScript-Bereich eine Funktion und rufen hinter dem Gleichheitszeichen diese Funktion auf, also z.B. `onClick="Funktion()"`.

<code>onAbort</code>	(bei Abbruch)
<code>onBlur</code>	(beim Verlassen)
<code>onChange</code>	(bei erfolgter Änderung)
<code>onClick</code>	(beim Anklicken)
<code>onDoubleClick</code>	(bei doppeltem Anklicken)
<code>onError</code>	(im Fehlerfall)
<code>onFocus</code>	(beim Aktivieren)
<code>onKeyDown</code>	(bei gedrückter Taste)
<code>onKeyPress</code>	(bei gedrückt gehaltener Taste)
<code>onKeyUp</code>	(bei losgelassener Taste)
<code>onLoad</code>	(beim Laden einer Datei)
<code>onMouseDown</code>	(bei gedrückter Maustaste)
<code>onMouseMove</code>	(bei weiterbewegter Maus)
<code>onMouseout</code>	(beim Verlassen des Elements mit der Maus)
<code>onMouseover</code>	(beim Überfahren des Elements mit der Maus)
<code>onMouseUp</code>	(bei losgelassener Maustaste)
<code>onReset</code>	(beim Zurücksetzen des Formulars)
<code>onSelect</code>	(beim Selektieren von Text)
<code>onSubmit</code>	(beim Absenden des Formulars)
<code>onUnload</code>	(beim Verlassen der Datei)
<code>javascript:</code>	(bei Verweisen)

Als Beispiel für die Verwendung von einem Event-Handler in einem HTML-Dokument sehen Sie hier den bekannten Rollover-Effekt, der Wechsel des Bildes sobald sich die Maus darüberbewegt:

```

```

Für diesen Effekt müssen Sie nicht JavaScript im Head-Bereich einbinden. Es ist aber ratsam, das zweite Bild mit folgendem Befehl vorzuladen, damit es ohne Verzögerung angezeigt werden kann.

```
<script type="text/javascript">  
var a = new Image();  
a.src="bild2.jpg";  
</script>
```

Ein weiteres Beispiel ist die Änderung Statuszeilen-Anzeige bei einem Link:

```
<a href="help.html" onmouseover="window.status='Hier erhalten Sie  
Hilfe'; return true;">Brauchen Sie Hilfe?</a>
```

Übersicht der Operatoren in JavaScript

Operation	Symbol	Bedeutung	Beispiel	Wert
Die mathematischen Operatoren				
Addition	+	Gibt die Summe zweier Zahlen zurück	<code>alert(8+5)</code>	13
Subtraktion	-	Gibt die Differenz zweier Zahlen zurück	<code>alert(8-5)</code>	3
Multiplikation	*	Gibt das Produkt zweier Zahlen zurück	<code>alert(8*5)</code>	40
Division	/	Gibt den Quotienten zweier Zahlen zurück	<code>alert(8/5)</code>	1.6
Modulo	%	Ermittelt den Rest der ganzzahligen Division zweier Zahlen	<code>alert(8%5)</code>	3
Inkrement	++	Erhöht eine in einer Variablen gespeicherten Zahl um 1	<code>X=4; x++; alert(x)</code>	5
Dekrement	--	Vermindert eine in einer Variablen gespeicherten Zahl um 1	<code>X=4; x--; alert(x)</code>	3
Die Vergleichsoperatoren				
gleich	==	Stimmen beide Werte überein?	<code>alert(8==5)</code>	false
ungleich	!=	Sind beide Werte unterschiedlich?	<code>alert("Wut"!="Mut")</code>	true
kleiner/gleich	<=	Ist der erste Wert kleiner oder gleich dem zweiten Wert?	<code>alert(8<=5)</code>	false
kleiner als	<	Ist der erste Wert kleiner als der zweite Wert?	<code>alert("Tag"<"S")</code>	false
größer/gleich	>=	Ist der erste Wert größer oder gleich dem zweiten Wert?	<code>alert(8>=5)</code>	true
größer als	>	Ist der erste Wert größer als der zweite Wert?	<code>alert("Wa">"Wau")</code>	false
Die logischen Operatoren				
UND	&&	Stimme beide Vergleiche überein?	<code>alert(8>5 && "a"=="z")</code>	false
ODER		Stimmt mindestens einer der Vergleiche überein?	<code>alert(8<5 8!=5)</code>	true
NICHT	!	Umkehr des Wahrheitswertes	<code>alert(!(8==5))</code>	true

Wichtig:

- Um sich von den Operatoren nicht einschüchtern zu lassen, ist es sinnvoll, sich zuerst den zu untersuchenden Sachverhalt auf einem Papier mit eigenen Worten zu notieren.
- Die Methode `alert()` ist nur ein Beispiel. Die Operatoren finden bei sämtlichen Anweisungen und Methoden Anwendung.
- Für Dezimalzahlen wird in JavaScript immer amerikanisch der Punkt verwendet (1.6) nicht das Komma (1,6).
- Bei Vergleichen von Strings geht es zuerst nach dem Alphabet dann nach Zeichenlänge.
- JavaScript arbeitet immer von oben nach unten, von links nach rechts. Ist beim Operator UND z.B. der erste Vergleich schon falsch, wird der zweite gar nicht mehr überprüft.