Ausarbeitung zu CSS & DHTML

1. CSS

1.1. Was ist CSS
1.2. Kompatibilitätsprobleme
1.3. Schematischer Aufbau der Stylesheets
1.4. CSS-Formate erstellen
1.5. Verschiedene CSS Attribute
1.6. Webeditoren und CSS

2. DHTML

2.1. „Rückblick“ HTML
2.2. Dynamic HTML = HTM dynamisch?
2.3. Die HTML Struktur
2.4. Das D.O.M.
2.5. Event-Handling: Ereignisse provozieren
Reaktionen
2.6. HTML: selbst programmieren oder einbauen
2.7. kleine Tricks -> Beispiele
2.8. Probleme der Lesbarkeit
->HTML und DHTML

3. Quellenangabe

1.1. Was ist CSS

CSS ist eine Spache, bestimmt für die Formatierung von HTML-Seiten. Sie greift auf die Sprachelemente von HTML zu und verleiht ihnen spezielle gewünschte Darstellungsmerkmale. Dies kann durch einmalige Vorschrift für ganze Sites oder auch bei einzelnen Tags erfolgen.

Das wichtigste Merkmal ist die Trennung von Inhalt und Präsentation.

1996 wurde CSS in der Version 1.0 vom W3-Konsortium als Standard empfohlen, seid 1998 gibt es eine Version 2.0. CSS bietet grundlegende Vereinfachung der Formatierung sowie erweiterte Möglichkeiten.

Cascading bedeutet, das mehere Formate auf einer HTML-Seite benutzt werden können. Der Webbrowser richtet sich nach einer gewissen Hierarchi, welche als „Kaskade“ bezeichnet wird, während er die Daten interpretiert.

Style Sheets, auf deutsch Formatvorlagen dienen der einheitlichen Gestaltung professioneller Web-Seiten.

1.1.1. Vorteile

- Die Webseiten, vor allem größere Päsentationen lassen sich wesentlich besser pflegen
- CSS ist von Mircosoft als Standard definiert
- Es ist moeglich globale Festlegung für Farben, Schriften, Hintergründe, Positionen u.s.w. für einzelne Seiten bis hin zu umfangreichen Dokumentsammlungen zu treffen.
- Man kann alle Formatierungsanweisungen in einer speziellen Datei mit der Kennung ".css" ablegen, auf die alle Seiten einer Präsentation zugreifen. Ändert man diese Datei dann hat dies Auswirkung auf die gesamte Präsentation.
- CSS bietet Formatieroptionen, die weit über die Möglichkeiten von HTML hinausgehen.
- Weil im HTML-Code Formatieranweisungen entfallen können, werden die Seitentexte erheblich kürzer, was der Übersicht und der Übertragungsgeschwindigkeit zugute kommt.
- Zukunftsorientiertheit durch Standardkonformität; nicht zuletzt werden analoge Verfahren der Differenzierung von Struktur und Stil (XML/XSL) im gesamten Publikationswesen und darüber hinaus an Bedeutung weiter zunehmen;

1.1.2. Nachteile

- Da bei CSS die Definitionen zentral getätigt werden, folgt daraus, daß bei einem eventuellen Fehler die ganze Präsentation diesen Fehler enthält
- Der „Programmierer“ muß eine neue „Sprache erlernen
- Der CSS-Standard ist noch nicht in allen Browsern vollständig integriert(siehe Kompatibilitätsprobleme.

1.2. Kompatibilitätsprobleme

Während beispielsweise Netscape 4 fast den vollen Sprachumfang der CSS-Version 1.0 interprtiert und einen Teil der Befehle der CSS-Version 2.0 , kennt der Internet Explorer die CSS-Version 1.0 bereits seit seiner Produktversion 3.0. In der 4. Version interpretiert er bereits einen Teil der CSS-Version 2.0 und einige spezielle, von Microsoft eingeführte Style-Sheet-Angaben.

Style-Sheets geben Ihnen zwar viele Möglichkeiten beim Webseiten-Layout. Jedoch sollte man beachten, dass die Layouts fehlerhaft wirken , falls die Anwender keine Style-Sheet-fähigen Browser benutzen. Da Netscape 3.x immer noch als wichtiger standard gilt, in dieser Version aber noch keine Style-Sheets kennt, werden Style-Sheets bislang noch vorsichtig eingesetzt. Gegen den Einsatz der Befehle der CSS-Version 1.0 ist eigentlich nichts mehr einzuwendenIn jedem Fall sollten Sie Ihre "gestylten" Web-Seiten immer auch mal mit Netscape 3.x testen.

- In frühren Netscape Versionen wurde CSS direkt von Java-Script interpretiert. Da viele Nutzer Java-Script jedoch aus Sicherheitsgründen deaktiviert haben, können die stylesheets nicht angezeigt werden.
- Probleme bei der Berechnung von Größen, da die Browser Pixel unterschiedlich umrechnen
- Unterschiedliches Verhalten der Browser bezüglich der Darstellungsmerkmale und Kompatiblität
-

1.3. Schematischer Aufbau der Stylesheets

Es gibt 3 Möglichkeiten CSS in HTML einzubinden: direkt in einem HTML-Tag, im Head-Bereich einer Website oder in einer externen Datei. Für einmalige Formatierung ist die Direktformatierung(direkt im Tag) von Vorteil, dateiweite Formatierung (im Head einer HTML-Datei) oder projektweite Formatierung (eigene CSS-Datei). Sinnvoll ist es jedoch weitestgehend alle Formatierungen in eine eigene Datei auszulagern. Dadurch erhält man zusätzliche Möglichkeiten, wie etwa browserabhängige CSS-Verwendung , oder die Benutzerauswahl unter mehreren CSS-Dateien . Die Pflege wird erheblich vereinfacht.

1.3.1. Direktformatierung

Um CSS-Formatierungen direkt auf einen einzelnen HTML-Tag anzuwenden, wird das Attribut style im HTML-Tag verwendet. Ein Beispiel:

<h1 style="color:red;">Tests</h1>

Durch diese Anweisung wird die Überschrift in diesem Fall rot angezeigt. Man muß also im öffnenden HTML-Tag zusätzlich das Style-Attribut, sowie eine entsprechende CSS-Angabe dahinter angeben.

1.3.2. Dateiweite Formatierung im Kopf-Bereich

Um dateiweite Angaben zu verwenden, wird im Kopfbereich einer HTML-Datei, also zwischen <head> und </head>, das Tag <style> notiert. Dahinter werden alle für diese HTML-Datei geltenden Angaben gemacht und dann wird der Bereich wieder mit </style> geschlossen. Alle hier gemachten Angaben gelten für die gesamte HTML-Datei. Ein Beispiel:

```
<style type="text/css">
<!--
p { font-family:Arial; }
//-->
</style>
```

Mit dem Pflichtattribut "type" wird der Style-Tag eingeleitet. Es muß also bei Verwendung von CSS in jedem Fall: <style type="text/CSS"> stehen. Die CSS Angaben sind in Kommentaren (<!-- //-->)gefaßt. Dies soll verhindern, dass sehr alte Browser die das Element <style> nicht kennen, versehentlich die CSS-Angaben als Ausgabe am Bildschirm anzeigen.

1.3.3. Einbindung externer Dateien

Um eine externe CSS-Datei mit einer HTML-Datei zu verbinden, gibt es 2 unterschiedliche Möglichkeiten. Die externe Datei kann innerhalb eines Style-Bereiches stehen, dies sieht dann beispielsweise folgendermaßen aus:

```
<style type="text/css">
<!--
@import url(style.css);
//-->
</style>
```

Durch die Syntax @import url(); wird die in der Klammer angegebene CSS Datei eingebunden. Diese Angabe lässt sich auch innerhalb einer CSS-Datei verwenden, um eine weitere CSS-Datei einzubinden. Netscape 4 versteht im Gegensatz zu fast allen anderen CSS-fähigen Browsern diese Syntax nicht. Daher lässt sie sich auch als Browserweiche verwenden, um Netscape 4 davon abzuhalten CSS-Befehle zu verwenden, die er mißinterpretieren würde.

Die HTML-Syntax zum Einbinden externer CSS-Dateien benutzt das Link-Tag. Ein Beispiel:

```
<link rel="stylesheet" type="text/css" href="style.css">
```

Das Attribut href gibt den Pfad zur CSS-Datei an, wobei auch hier wieder die allgemeinen Regeln für Pfade gelten. Zusätzlich sind auch die Attribute rel und type Pflichtattribute. Durch rel wird dem Browser mitgeteilt, dass es sich um ein Stylesheet handelt das eingebunden werden soll (das Element link kann auch für andere Zwecke verwendet werden), durch das Attribut type wird der Mime-Type angegeben.

Individuelle Browsereinstellungen können ebenfalls Einfluß nehmen.

1.4. CSS-Formate erstellen

Egalob man Formate zentral definiert oder aus einer separaten CSS-Datei einbindet, man kann einzelne HTML-Elemente formatieren.

Verschiedene Arten von Formaten

- Formate für Elemente
- Formate für verschachtelte Elemente
- Attributbedingte Formate
- Formate für Klassen
- Individualformate
- Pseudoformate

Will man also einem HTML-Tag ein bestimmtes Format zuweisen, folgen nachdem man diesen notiert hat in geschweiften Klammern die Formatdefinitionen.

Diese Definitionen werden aus der Formateigenschaft und einem nach einem Doppelpunkt folgenden Wert erstellt, dieser Wert wird der Formateigenschaft zugewiesen.Werden in diesem Klammergebilde mehrere Formateigenschaften einem Tag zugewiesen werden diese durch Semikolon voneinander getrennt.

Gennerelles Schema

Objekt

Klasse (Klassifizierung){Eigenschaft:Wert}
Body. Abs {color: blue}

Beispiele:

- b{ color : green; }
  b{ font-style : Times New Roman; }
  b{ font-size : 30px }
- Gruppierte Eigenschaften:
  b{ color : green;
  font-style : Times New Roman; }
  font-size : 30px; }
- Gruppierte Eigenschaften und Gruppierte Selektoren
  b, p, h3{ color : green;
  font-style : Times New Roman; }
  font-size : 30px; }
- Formate für Klassen definieren.

  Klassenvereinbarung:
  b.veryBigOne{ color: blue; }
  Anwendung dieser:
  <b class=veryBigOne> .....<b>

  Man kann auch ID's nutzen um Styles zu definieren und zuzuweisen:

- Definition der Styles für ID testStyle über das DoppelKreuz.
  Vereinbarung:
  #testStyle{ color: blue; }
  Anwendung dieser:
   ...

Weitere Bemerkungen:

- Vererbung:
  Tags, die auf andere aufbauen, erben ggf. auch deren Attribute. Sofern diese nicht überschrieben werden. Setzt man z.B: im "body" die Textfarbe auf blau, gilt dies fuer das gesamte Dokument. Ausser ein Tag ueberschreibt dies, dann gilt für diesen Bereich die lokale Definition.
- Kommentare in CSS Code:
  Durch /* Kommenar */ werden Kommentare beschrieben. Diese werden dann vom Browser ignoriert.

1.5. Verschiedene CSS Atribute

Es gibt im CSS 2 standard bereits deutlich über 100 Attribute von denen mehr als die Hälfte auch schon im CSS 1 vorhanden waren.

1.5.1 Schrifteigenschaften

font-family: "Courier New", Courier, monospace
font-style: italic
font-variant: small-caps /* Kapitälchen */
font-weight: bold;
font-size: 12pt

1.5.2. Farb- und Hintergrundeigenschaften

color: #FF0000
background-color: #FFFFFF
background-image: url("grafik.gif")
background-repeat: no-repeat
background-position: 5% 5%
background-attachment: fixed

1.5.3. Texteigenschaften

word-spacing: 1em
letter-spacing: 0.1em
text-decoration: underline
text-transform: capitalize
text-align: justifybackground-attachment
text-indent: 3em

line-height: 120%

1.5.4. Box-Eigenschaften

margin-top: 10px
margin-bottom: 1em
margin-right: 1em
margin-left: 0.5em
padding-top: 5px
padding-bottom: 5px
padding-right: 10px
padding-left: 10px
border-width: 1px
border-color: #808080
border-style: solid
width: 640px
height: 480px

1.5.5. Klassifikations-Eigenschaften

white-space: nowrap
list-style-type: circle
list-style-image: url(http://images.com/arrow.gif)
list-style-position: inside

In CSS ist es möglich Längenangaben absolut und relativ anzugeben.

absolut: Zoll (**in**), Zentimeter (**cm**), Millimeter (**mm**), Pixel (**px**), Punkt (**pt**) und Pica (**pc**).

Relativ: **px**, **em** und **ex** .

Die Angabe em bezieht sich auf die Größe der aktuell gültigen Schrift;

ex bezeichnet die Höhe des Kleinbuchstaben x der verwendeten Schriftart.

Die betreffenden Angaben dürfen von dem eigentlichen Werten nicht durch Leerzeichen getrennt sein. z.B. p.keywords { word-spacing: 1cm }

1.6. Webeditoren und CSS

Für die Entwicklung und Programmierung von **Stylesheets** lassen sich alle Text-Editoren verwenden. Komfortabler geht es natürlich mit speziellen Editoren wie Frontpage, **HTMLPad**, **Rapid CSS**, **TopStyle**, **StyleMaster**, **Simple CSS**, **CSSEdit** und **SkEdit** um nur einige zu nennen.
Hier gibt es aber auch wieder einige Qualitätsmerkmale die man beachten sollte.

- Allgemein
  Suchfunktionen, Undo-Möglichkeit, FTP Siteverwaltung, Integrationsfähigkeit/Schnittstellen, Validator-Integration, Sprache..
- Synax-Highlighting
  für CSS, HTML oder PHP
- Code-Vervollständigung
  für CSS, HTML
- Assistenten
  für Stylesheets, Navigationselemente oder Layouts

2. DHTML

2.1. HTML

Bevor wir zur DHTML kommen, möchte ich noch einmal HTML in Erinnerung rufen. HTML ist eine Seitenbeschreibungssprache im Web.
Das Grundgerüst sieht folgendermaßen aus:

```
<html>
<head>
  <title> ... </title>

  <meta ....>

</head>
<body>
  <h1> ... </h1>
  ....
  <h6> ... </h6>
  <p> ... </p>

  <img src=url height= width= alt= >

  <strong> ... </strong>
  <em> ... </em>
  <u> ... </u>

</body>
</html>
```

Kurz noch etwas zum Aufbau der Elemente, die später wichtig für DHTML sind:

<IMG SRC="Bild1.jpg">
<_tag__Attribut___wert____>

Bevor man mit DHTML ordentlich arbeiten kann, braucht man natürlich HTML Kenntnisse. Das Thema hier ist DHTML und die Grundlagen von HTML werden vorausgesetzt.

2.2. DHTML = HTML dynamisch?

HTML ist eine seitenbeschreibende Sprache, ist immer gleich aufgebaut, also statisch. Es werden einfach Zeilen interpretiert und wiedergegeben, d.h. keine Steuerung durch die Site. Der Besucher ist beschränkt und kann entweder auf Links klicken oder weiterlesen.

Man kann es mit einer Zeitung vergleichen, die man lesen kann und weiterblättern, aber es ist keine „Dynamik“ dahinter.

DHTML ist eine Zusammensetzung verschiedener Komponenten. Es beinhaltet weiterführende Programme. Diese sind in Tags geschrieben und rufen ggf. weitere Funktionen auf. Dadurch wird die Site attraktiver gestaltet. Es gibt hat eine Vielzahl an Möglichkeiten, eine punktgenaue Steuerung durch Programmieren zu beeinflussen.

2.2.1. Was ist DHTML?

HTML ermöglicht es, Informationen in sich abgeschlossene Häppchen zu zerteilen und über diese Informationseinheiten eine Verweisstruktur zu legen. Der Anwender ist nicht mehr gezwungen, wie bei anderen am Computer erstellten Texten eine vorgegebene Lesereihenfolge einzuhalten. Mit Querverweisen/Hyperlinks kann er das Angebot durchstöbern. Auch deshalb werden inzwischen viele Hilfedateien und Programmdokumentationen in HTML geschrieben.
Die meisten Anwender wollen jedoch mehr. Immerhin haben sie viel Geld investiert, um das Internet zu durchstöbern, und sie erwarten nun etwas mehr als die elektronische Wiedergabe eine Buchseite. Diese Erwartungshaltung führt direkt zu Dynamic HTML und einer Seite, die veränderbar ist nachdem der Server die Seite an den Browser gesendet hat.
DHTML ist keine klassische HTML-Erweiterung wie HTML 4.0 in Gestalt neuer Tags und auch keine neue Sprache. DHTML ist eine Kombination von mehreren Techniken, unter anderem HTML, Cascading Style Sheets, Scripting und objektorientierter Programmierung, die zusammen eingesetzt werden, um Webseiten zu gestalten.
Dynamic HTML erlaubt es dem Entwickler von Webseiten, die Darstellung einer Seite zu ändern, indem er ein Style Sheet anpasst und indem er auf die Eigenschaften, Methoden und Ereignisse für die HTML-Elemente zugreift, um diese Elemente dynamisch oder statisch zu positionieren oder ihr Erscheinungsbild zu ändern.

2.2.2. Vorteile und Nachteile von DHTML

DHTML bringt einige neue Features zur Webseitengestaltung. Einige Beispiele werden Sie hier kennen lernen. Eine intensive Anwendung von DHTML bringt nicht nur Vorteile mit sich, es existieren auch einige Einschränkungen.

2.2.2.1. Vorteile und Möglichkeiten

- Die dynamische Änderung des Inhaltes nach dem Laden der Seite
- Pixelgenaue Platzierung von Inhalten ohne blinde Tabellen oder GIFs
- Einbindung multimedialer Inhalte wie Shockwave in Container
- Wechselnde Inhalte ohne Nachladen der Seite
- Die Stiländerung der Seite durch den Benutzer
- Die dynamische Änderung von Textattributen
- Pop-Ups für Links mit Erklärungen und Hinweisen

2.2.2.2. Nachteile und Einschränkungen

- Die Bildschirmgröße des Users muss bekannt und alternative Seiten sollten vorhanden sein.
- Es existiert das Problem der nachträglichen Veränderung der Fenstergröße durch den Benutzer.
- Die beiden Browser haben eine unterschiedliche Fenstergröße im Hinblick auf die genaue Inhaltspositionierung mit den Style-Attributen 'left' und 'top'.
- Es sollte schon eine Bildschirmauflösung von mind. 800x600 beim User vorhanden sein und das Verwenden der 4er-Versionen der Browser ist obligatorisch.
- Das Ausdrucken von Inhalten in versteckten Containern ist nicht möglich.
- Ein absolut sauberer Code für beide Browser ist notwendig, alle Tags müssen abgeschlossen werden. Der kleinste Fehler führt zu katastrophalen Ergebnissen.

2.3. Die Struktur und Komponenten von DHTML

Nach der Meinung von Netscape und Microsoft soll DHTML neben seinen dynamischen Aspekten auch die Webseiten-Präsentation und die statische Positionierung beinhalten.

Die wichtigsten Komponenten von DHTML sind:

- Cascading Style Sheets 1/2 (CSS) und JavaScript Accessible Style Sheets (JASS) zur Webseiten-Präsentation
- Das Bereitstellen von HTML-Elementen für das Scripting
- Das dynamische Ändern des Erscheinungsbildes von HTML-Elementen
- Das Verbergen, Anzeigen, Verschieben und Zuschneiden von HTML-Elementen
- Das Auffangen von Ereignissen für traditionelle Elemente
- Multimedia-Erweiterungen wie Übergangseffekte, Filter und Dynamic Fonts. Einige dieser Technologien haben beide Browser gemeinsam, andere sind proprietäre Eigenschaften von Microsoft und Netscape. Als CrossBrowser-DHTML wird die Anpassung an die Eigenschaften beider 4er-Browser bezeichnet. Der Begriff wurde von Kevin Lynch geprägt.

Folgende DHTML-Konzepte unterstützen beide Browser gemeinsam, obwohl sich ihre spezifische Unterstützung im Detail unterscheidet:

- Cascading Style Sheets
- Das Document Object Model
- JavaScript

Folgende Features sind browserspezifisch und arbeiten nicht mit dem jeweils anderen Browser zusammen:

- <layer>-Tag
- JavaScript Accessible Style Sheets
- Bitstream Fonts
- Direct Animation Controls
- Data Binding
- VBScript
- OpenType Fonts

2.4. Das Document Object Model

Ziel von DHTML ist es, alle Elemente einer Webseite dynamisch veränderbar zu machen. Dazu muss mit einer Scriptsprache wie JavaScript gezielt auf bestimmte HTML-Elemente zugegriffen werden können.

Das DOCUMENT OBJECT MODEL versucht, alle relevanten Bestandteile einer angezeigten Webseite als eine Objekthierarchie abzubilden. Diese Hierarchie sollte sich dann in objektorientierten Programmiersprachen wie JavaScript wieder finden.

Das W3C hat einen Vorschlag zu einem Modell veröffentlicht. In der aktuellen HTML-Version 4.0 ist festgelegt, welche HTML-Befehle welche Event-Handler wie 'mouseOver' zum Steuern von Ereignissen haben können. Das DOM ist zwar ein Konzept beider Browser, aber wie immer sind die Ansätze von Netscape und Microsoft verschieden, eigentlich sogar entgegengesetzt, wobei Microsoft das DOM weitgehend umgesetzt und Netscape für die 5er-Version Besserung versprochen hat.

Um Scripts zu schreiben, die mit beiden Browsern funktionieren, ist es z.Z. noch notwendig jedes Ereignis als einzelne Funktion zu behandeln. Man muss sich also an die strenge Netscape-Objekthierarchie halten, da diese auch vom Explorer unterstützt wird.

2.4.1. Das Modell von Microsoft

Vom Internet-Explorer wird der Scriptzugang zu allen HTML-Seitenelementen unterstützt, einschließlich der Style Sheets. Seitenelemente werden als Objekte enthalten in einer document.all-Sammlung erfaßt und können durch Index, Name oder ID angesprochen werden.

Beispiel: den Namen aller Tags einer Seite schreiben

```
for (i=0;i<document.all.length;i++)
    {document.write(document.all[i].tagName + "\n";}
```

Der IE benutzt dazu ein 'Event bubbling model':
Aktionen, die von einem hierarchisch niedrigeren Objekt kommen, werden an die höherwertigen Elemente weitergereicht (Luftblasen-Prinzip). Jedes Seitenelement kann Ereignisse erzeugen und ausführen. Jeder simple HTML-Tag wie <H1> oder <BLOCKQUOTE> kann so mouseOver-, mouseOut- oder onClick-Ereignisse ausführen.

Ein Trick, um ähnliches mit dem Navigator zu erreichen, ist, das Element in ein leeres ANCHOR-Tag einzuschließen. Ein leeres ANCHOR-Tag verweist nicht auf eine URL oder NAME, wie der folgende Code darstellt:

```
<H1><A HREF="" onClick="eine_function();return false">Die Überschrift</A></H1>
```

Um die Farbe und die Textdekoration für den Anker-Link zu entfernen, wird dem ANCHOR-Tag ein Style Sheet zugewiesen: A {text-decoration: none; font-color: wie Text}.

Für den Ereignis-Handler von 'onClick' muß' der Wert 'false' zurückgegeben werden, sonst wird das Ereignis von dem Anker-Element verarbeitet. Weil das Dokument auf eine leere URL verweist, wird dann das Verzeichnis angezeigt, indem das Dokument enthalten ist.

2.4.2. Das Modell von Netscape

Von Netscape wird der Scriptzugang nur zu einer spezifischen Anzahl von Seitenelementen unterstützt, z.B. den Layern einer Seite. Layer im Navigator beinhalten einen Seitenabschnitt/Container, begrenzt vom LAYER-Tag und die Positionierung durch CSS-Attribute.

Der Navigator benutzt das sogenannte 'Capturing model', das Gegenteil des 'Event bubbling models'. Dies besagt, dass die Aktionen von Mutter-Objekten zu Tochter-Objekten durchgereicht werden. Die Objekte, mit denen der User interagieren kann, sind streng limitiert. Es sind: Formulare, Plug-Ins, Frames, Anchors, Links, Layers, Applets und Bilder.
Statisch hingegen bleiben die restlichen Elemente. Die speziellen Elemente können durch Name, ID oder Index angesprochen werden.

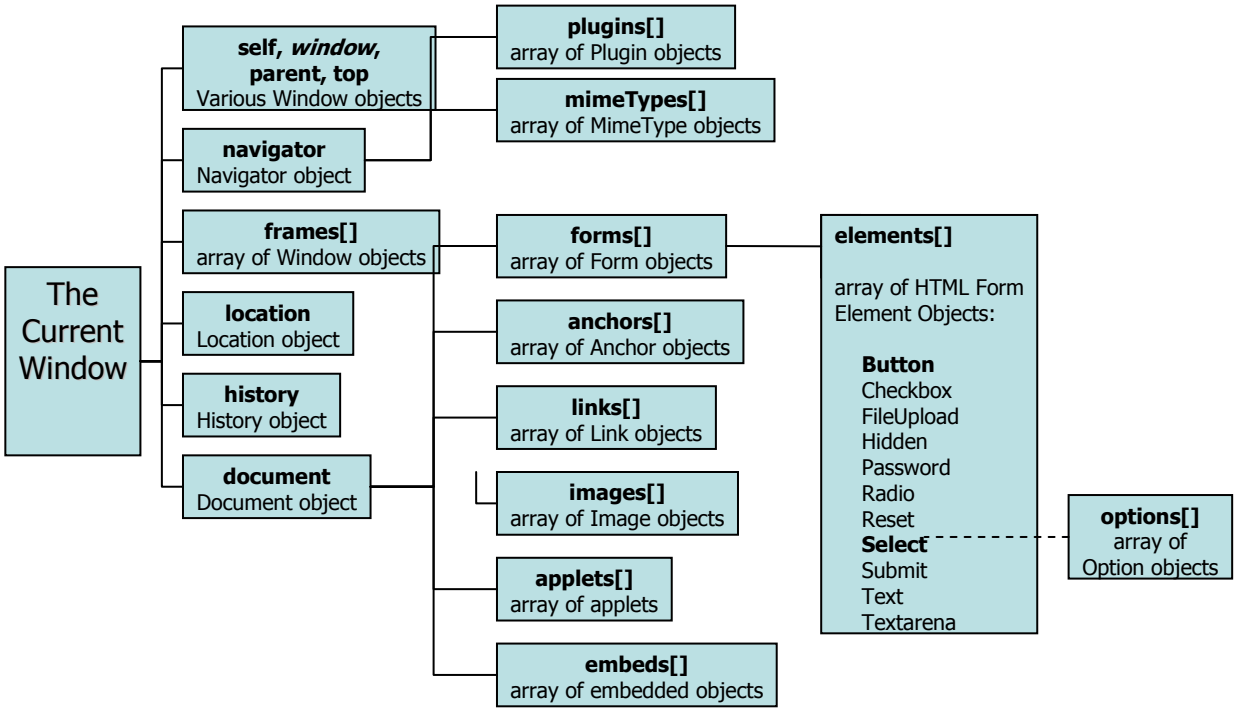
Beispiel: den Namen aller Layer einer Seite schreiben

```
for (i=0;i<document.layers.lenght;i++)
    {document.write(document.layers[i].name +"\n";}
```

Bei beiden Browsern erscheint jede Änderung der Objekteigenschaften augenblicklich auf der Seite.

DOM Struktur

(http://www.w3.org/TR/REC-DOM-Level-1/ecma-script-language-binding.html)



2.5. Event-Handling – Ereignisse provozieren Reaktionen

Event-Handling bedeutet zu Deutsch: Ereignis-Behandlungsprozeduren

Im Grunde gibt es viele verschiedene Event-Handler, die man miteinander kombiniert und neue draus macht.
Zwanzig häufig benutzte sind folgende:

OnAbort	OnError	OnMouseover	OnReset	OnKeyup
OnBlur	OnFocus	OnMousemove	OnSelect	OnLoad
OnChange	OnKeydown	OnMouseup	OnSubmit	OnMousedown
OnClick	OnKeypress	OnMouseout	OnUnload	OnDblClick

2.5.1 Demonstrative Vorstellung einiger Event-Handler vorstellen:

2.5.1.1. OnClick

```
<html>

<head>
<meta http-equiv="Content-Language" content="de">
<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>kleines beispiel zum OnClick</title>

<link rel="sylesheet" href="style.css">
</head>

<body class=center>


<table class=center border="1" cellpadding="2" cellspacing="2"  width="80" >
  <tr>
    <td id="eingabe"
onMouseover="document.all.eingabe.innerText='und nun die Maus Taste...'"
onclick="document.all.eingabe.innerText='Danke...'"
onMouseout="document.all.eingabe.innerText='Bitte hier klicken:'" width="100%">Bitte
klicken sie hier</td>

  </tr>
</table>
</body>

</html>
```

Event-Handler werden direkt am Tag hinzugefügt. Für den Tag <p> würde der Event-Handler OnClick also so notiert:

<p onclick="…" >

Tritt das Ereignis ein, verweist der Event-Handler auf dir in anführungszeichen stehenden Attribut. Es könnte hier noch eine abfrage stehen, ein JavaScript usw.

2.5.1.2. OnBlur

```
<html>
<head>
<script type="text/javascript">
```

```
function message()
{
alert("Zurück bitte, sie haben Ihren Namen nicht eingegeben!")
}
</script>
</head>

<body>

<p>

Der OnBlur-fall tritt auf, wenn ein Element Fokus verliert.  Versuchen Sie, auf dem Eingangsgebiet zu klicken oder zu schreiben, dann klicken Sie irgendwohin im Dokument, also verliert das Eingangsfeld Fokus
</p>
<form>
Hier Name eingeben: <input type="text" onblur="message()" size="20">
</form>

</body>
</html>
```

Diese kleine Routine prüft das Eingabefeld. Hat der User einmal die Eingabe vergessen, wird er mit einer Meldung hingewiesen.

2.5.1.3. OnMouseover & OnMouseOut

```
<html>
<body>

<h1 onmouseover="style.color='red'"
onmouseout="style.color='black'">
Mouse over this text</h1>

</body>
</html>
```

Ist der Mauszeiger einmal über diese Überschrift <h1> wird sein Text aus Schwarz zu Rot.

2.5. DHTML: Selbst Programmieren oder einbauen?

Zunächst stellt sich die Frage, welche Hilfsmittel zur Erstellung der Webseiten mit DHTML-Features notwendig sind. Für die Erstellung von Webseiten mit eingebauten DHTML-Funktionen benötigen Sie eigentlich nur eins: einen Texteditor.

2.5.1. Einbauen?

Das Programm Dreamweaver bietet alle notwendigen Funktionen an, die zur schnellen Entwicklung von Seiten notwendig sind.
Grundsätzlich ist aber jeder Webeditor geeignet, der zu den üblichen Funktionen auch die Bearbeitung des HTML-Codes durch den Entwickler zulassen. So lassen sich die Seiten genauso im Microsoft FrontPage 2000 erstellen.
Beide Programme stellen zwar DHTML- und vorgefertigte JavaScript- bzw. JScript-Routinen zur Verfügung, diese sind aber nicht ausreichend.

2.5.2. Programmieren?

Grundvoraussetzung für die eigene Programmierung ist zunächst einmal die grobe Kenntnis der Programmiersprachen, die mit den Event-Handler-Funktionen aufgerufen werden. Hier kommen Sie um das die entsprechenden Kapitel und weiterführende Informationen nicht drum herum.
Neben JavaScript bietet sich per DHTML jedoch die Möglichkeit, eine Vielzahl von anderen Programmiersprachen einzusetzen, z.B. Java-, Perl-, ActiveX- und Visual Basic-Routinen.

2.6. DHTML-Tricks aus dem WWW

2.6.1. Tipp

Natürlich gibt es das eine oder andere Script im World Wide Web, das man einbauen kann. Befragt man eine Meta Suchmaschine nach einen Begriff wie z.B. „JavaScript“ geben parallel andere Meta Suchmaschinen eine ausführliche Liste der gefundenen Information.

In Deutschland bietet sich die Suchmaschine MetaGer der Universität Hannover geradezu an. Sie sucht nicht nur die Daten sehr verlässlich heraus, sondern prüft auf Wunsch auch, ob die Seiten noch verfügbar sind. Adresse: http://meta.rrzn.uni-hannover.de

2.6.2. CrossBrowser DHTML

Es gibt einige Tricks, mit deren Hilfe Sie browserübergreifende Seiten erstellen können. Sowohl Netscape als auch der IE unterstützen den DIV-Block und beide unterstützen die statische CSS-Positionierung für diese Elemente. Sie sollten also besser den DIV-Tag für Container benutzen statt dem LAYER-Tag, zumal auch Netscape die Verwendung von Layern inzwischen nicht mehr empfiehlt.

Der IE beinhaltet mehr Ereignisse für HTML-Elemente als der Navigator, aber durch die

Verwendung des schon beschriebenen leeren ANCHOR-Tags, können die Möglichkeiten beider Browser angeglichen werden.

Darüber hinaus ist noch das Anlegen separater Codeblöcke möglich oder Prüfungen innerhalb eines Codeblocks.

Um Objekte in beiden Browsern ansprechen zu können, wird eine Referenz abgefragt wenn die Seite geladen wird, um zu erfahren welcher Browser vom User genutzt wird. Dazu wird LAYER von Netscape benutzt, da der IE diesen Tag nicht kennt.

```
if (document.layers){ns=1; ie=0;}
else {ns=0; ie=1;}
```

Es wird überprüft, ob der benutzte Browser das LAYER-Objekt kennt. Wenn ja, wird der Variablen ns für den Navigator das Flag 1 für 'true' und der Variablen ie das Flag 0 für 'false' zugewiesen. Wird das Objekt nicht erkannt, ist die Zuweisung umgekehrt.
Um die ermittelten Werte weiter verwenden zu können, wird eine Funktion zur Initialisierung des geladenen Dokumentes benötigt:

```
function init()
{if (ns) ebene=document.nameEbene;
if (ie) ebene=nameEbene.style;}
```

Diese Funktion wird im BODY-Tag mit 'onLoad' eingebunden:

```
<BODY onLoad="init()">
```

Auch die Verwendung folgenden Crossbrowser-APIs von Macromedia ist möglich und sinnvoll:

```
if (navigator.appName=="Netscape")
     {layerRef="document.layers";styleRef="";}
else {layerRef="document.all";styleRef=".style";}
```

Im Prinzip geht es immer nur um die Abfrage des verwendeten Browsers und um die Umsetzung des Ergebnisses in <wenn Netscape, dann document.layers nutzen; wenn IE, dann document.all verwenden>, dh. die Berücksichtigung der verschiedenen DOM-Ansätze beider Browser.

Wie Sie sicher schon inzwischen gemerkt haben, werden Sie nicht daran vorbeikommen, sich mit JavaScript zu beschäftigen, wenn Sie DHTML einsetzen wollen.

2.7. Probleme der Lesbarkeit

2.7.1. Fremde Programmierung

Fremde Programmierung hat einen großen Nachteil: man muss diese Programmierung erst mal erlernen. Manche Programme bieten Komplexe Funktionen an, diese müssen nicht unbedingt leichter zu verstehen, einfach einzubinden oder auf den Server lauffähig sein. So entstehen Fehler auf der Seite.

2.7.2. Editoren

So bietet z.B. Microsoft FrontPage 2000 so manches Script an, das allerdings nur auf den Servern arbeitet, die die Microsoft Server Software besitzen.
Andere Webeditoren bieten Scripte an die Lauffähig sind, aber auch umfangreich sind. So haben Sie z.B. in HotDog Professional – einem sehr guten HTML-orientierten Webeditor – die Möglichkeit, auf mitgelieferte Scripte zurückzugreifen.

2.8. Lösungen bei DHTML-Problemen

Vermehrt treten Probleme auch wenn mit DTHML arbeitet. Grund dafür sind kleine Abweichungen in der Syntax, allgemeine Tippfehler oder Fehler in fremden Scripten die man versucht zu integrieren. Hier sieht man, dass man gute Programmierkenntnisse braucht auch wenn man Scripte einbauen möchte.
Aber auch Browserspezifische Besonderheiten spielen da eine große Rolle.

Was man bedenken sollte, um Fehler zu vermeiden:

- Man sollte mit unterschiedlichen Browsern rechnen
- Alle Browser müssen dieselbe Funktionalität erhalten
- Wir senden allen nicht-DOM-kompatiblen Browsern eine nur-Text-Variante der Site
- Browser von Version 4.0 oder höher erhalten dynamische Version, alle andern müssen lesen und navigieren können

3. Quellenangabe:

- WWW.w3.org
  - Das w3.org-Konsortium
- Das große Buch (x)HTML & XML
  - 1. Auflage 2000 by DATA BECKER (F. Harms, D. Koch, O. Kürten)
  ISBN: 3-8158-2105-3
- http://de.selfhtml.org