Seminararbeit zur Wirtschaftsinformatik SS 2001

D-HTML

Die neuen Möglichkeiten des DHTML im Vergleich zum herkömmlichen HTML

Von Piero Musio

Im Rahmen des Blockseminars zur Wirtschaftsinformatik

von Prof. Markus Lusti

am 30./31. Mai 2001

Inhaltsverzeichnis

Einleitung

Im Rahmen des Seminars zur Wirtschaftsinformatik möchte ich das „Dynamic HTML“ (abgekürzt DHTML) vorstellen und vor allem die neuen Möglichkeiten des DHTML im Vergleich zum herkömmlichen HTML erarbeiten.
Kritiker sagen, dass das dynamische HTML eine Erfindung von Marktstrategen sei. Ist DHTML nun wirklich eine völlig neue Sprache zur Gestaltung von Webseiten oder wurde das altbekannte HTML einfach um einige Befehle erweitert?
Im ersten Kapitel meiner Arbeit stelle ich zuerst kurz vor, was mit HTML alles möglich ist, um Webseiten zu gestalten. Ich beschreibe aber nicht jeder einzelne HTML-Befehl, sondern erkläre nur grob die Grundidee bei der HTML-Programmierung. Weiter kommt eine Aufzählung der einzelnen Webseitenelemente, die mit HTML darstellbar sind.
Im zweiten Teil präsentiere ich das DHTML, was es ist und welche seine Grundkomponenten sind. Als Grundlagen des DHTML zählen:

- HTML
- Formatvorlagen (Style Sheets)
- Skriptsprachen (JavaScript)

HTML erwähne ich schon im ersten Teil. Im Kapitel Formatvorlagen erkläre ich kurz wie man Style Sheets alles definieren kann. Skriptsprachen sind ein bisschen schwieriger zu verstehen, darum habe ich dieses Thema ausführlicher behandelt. Hier schreibe ich über Funktionen, Event-Handler und Objekte. Dann kommt ein kurzes Kapitel über Fenster und Frames, das genauso gut unter HTML stehen könnte.
Im dritten Kapitel zeige ich nun ein konkretes Beispiel einer DHTML-Seite. Hier stossen wir auf ein Problem. Wegen der unterschiedlichen Entwicklung der Version 3.0 auf der Version 4.0 von Microsoft Internet Explorer und Netscape Communicator ist deren DHTML-Syntax nicht genau gleich. Ich zeige also ein konkretes Beispiel zuerst mit Microsoft und dann mit Netscape. Will man DHTML so programmieren, dass es sowohl auf Microsoft Internet Explorer ab Version 4.0 als auch auf Netscape Communicator ab Version 4.0 läuft, gibt es dafür eine Cross-Browser-Lösung. Als letztes Unterkapitel beschreibe ich kurz einige weitere Effekte, die mit DHTML möglich sind.

1 Möglichkeiten mit HTML

HTML bedeutet „**H**yper**t**ext **M**arkup **L**anguage“. Dabei bedeutet Markup Language, dass es sich um eine Auszeichnungssprache handelt. Als solche enthält HTML Befehle zum Markieren typischer Elemente eines Dokuments, wie Textabschnitte, Überschriften, Absätze, Listen, Tabellen und Grafikreferenzen.
Das vorangestellte „Hypertext“ steht für die wichtigste Eigenschaft von HTML. Auf einer HTML-Seite können nämlich Verweise stehen, die entweder zu anderen Stellen auf der gleichen Webseite oder zu einer Webseite auf einen räumlich weit entfernten Rechner führen. Somit ist es möglich alle Rechner der Welt zu vernetzen, was die Grundidee des World Wide Webs ist.

HTML beschreibt also Dokumente. Dabei sind die Beschreibungsbefehle hierarchisch gegliedert. Dokumente haben zum Beispiel eine Hintergrundsfarbe, einen Titel und einen Text. Diese Elemente werden in HTML der Reihe nach aufgezählt und durch Kurzbefehle abgegrenzt. Diese Kurzbefehle, die sog. *tags*, werden in spitzen Klammern geschrieben und beschreiben nichts anderes als Anfang und Ende eines Textelements. Zum Beispiel wird eine Überschrift 1. Ordnung ganz einfach so dargestellt:

```
<h1>Überschrift 1. Ordnung</h1>
```

Tags stehen meistens paarweise: <h1> beschreibt den Anfang der Überschrift und </h1> (gesprochen: End-h1) das Ende.
Einige Textelemente können auch Unterelemente enthalten. Ein Dokument kann einen Absatz mit einer fett markierten Stelle, eine Aufzählung mit einzelnen Listenpunkte oder eine Tabelle mit Einzelzellen enthalten. Solche Unterelemente werden einfach hierarchisch am Oberelement angehängt. Beispielsweise wird eine Liste so geschrieben:

```
<ul>
<li>1. Listenpunkt</li>
<li>2. Listenpunkt</li>
</ul>
```

 heisst „**u**nordered **l**ist” und markiert den Anfang einer nicht nummerierten Aufzählung. Die meisten tags sind nichts anderes als die Abkürzungen der entsprechenden englischen Wörtern. Mit ein bisschen Englischkenntnissen ist nicht schwierig zu erraten, dass <li> „**l**ist **i**tem“ bedeutet und ein Listenpunktelement markiert.

HTML-Seiten können auf jedem beliebigen Texteditor geschrieben werden. Dabei muss die Datei mit der Endung „.htm“ oder „.html“ abgespeichert werden. Nun ist es Aufgabe des Browsers die HTML-Datei am Bildschirm anzuzeigen. Die beschriebenen Textelemente erscheinen dann, natürlich ohne die entsprechen tags, optisch gut erkennbar und strukturiert im Browserfenster.

HTML ist nicht „case-sensitive“, Gross-/Kleinschreibung spielt also keine Rolle und HTML-Quelltexte können in Grossbuchstaben oder in Gross-/Kleinschreibung eingegeben werden.

Ein grosser Vorteile des HTML im Gegensatz zum DHTML ist, dass HTML in allen Browsern läuft sowohl auf Microsoft Internet Explorer als auch auf Netscape Navigator. In

DHTML ist bei der Programmierung zwischen den zwei Browsern zu unterscheiden. Darauf werde ich aber später noch genauer eingehen.

1.1 Grundstruktur einer HTML-Datei

Die ganze HTML-Datei wird vom tag-Paar <html> und </html> abgegrenzt, welches Anfang und Ende der Webseite markiert. Innerhalb dieses tag-Paar wird der head-Teil und der body-Teil geschrieben.

- Im head-Teil stehen die Angaben, die für die Verwaltung der HTML-Datei notwendig sind, wie Titel und Autor. Diese Angaben erscheinen in der Kopfleiste des Browserfensters und werden für Bookmarks und Suchhilfen verwendet.
- Im body-Teil stehen die eigentlichen Informationen zum Aufbau der Webseite, wie Textelemente, Grafiken, Töne und Links.

```
<html>

<head>
<title>Titel der Webseite</title>
</head>

<body>
... Hier kommt der Inhalt der Webseite ...
</body>

</html>
```

1.2 Textelemente

Wie schon erwähnt kann man mit HTML verschieden Textelemente darstellen. Dazu gehören im lediglichten:

- **Überschriften** 1. bis 6. Ordnung, welche sich in der Grösse und im Zeilenabstand unterscheiden.
- **Absätze**, die eine Zeilenschaltung verursachen. Absätze können zentriert, rechts- oder linksbündig ausgerichtet werden.
- **Textformatierung**, um Fettschrift bzw. Kursivschrift zu erzeugen.
- **Textabschnitte**, die im Textfluss ausgerichtet werden können. In einem linksbündigen Text kann zum Beispiel einen rechtsbündigen Abschnitt definiert werden.
- **Aufzählungen**, die geordnet oder ungeordnet sein können. Eine geordnete Aufzählung ist nummeriert eine ungeordnete benutzt lediglich ein Aufzählungssymbol. Dieses kann ein hohles Quadrat, ein gefüllter oder hohler Kreis sein.
- **Tabellen** ohne oder mit Rahmenlinien unterschiedlicher Breite. Tabellen bestehen aus einem Tabellenkopf und einem Tabellenkörper. Auch können verbundene Zellen gebildet werden, die über mehrere Spalten reichen. Innerhalb einer Tabellenzelle können dann zusätzlich die Textelemente verwendet werden, die ich bis jetzt aufgezählt habe.

All diese Elemente und den dazugehörenden Hintergrund können mit unterschiedlichen Farben dargestellt werden. Das gleiche gilt auch für den Hintergrund der ganzen Seite. Dabei können theoretisch über 16 Millionen Farben dargestellt werden. Davon zählen 16 als Standardfarben und ca. weitere 140 als vordefinierte Farben. Für die Definition der Farben in HTML-Programmierung gibt es verschiede Schreibweisen auf die ich jetzt nicht tiefer eingehen möchte.

Auch können Webseiten sog. „Links" (to link: verbinden, verknüpfen) enthalten. Diese sind nichts anderes als anklickbare Wörter oder Textabschnitte auf der Webseite. Klickt man auf einen solchen Link, dann kommt man von einer Stelle auf der Webseite zu einer anderen Stelle auf der gleichen Webseite. Auch kann man von einem Link aus auf einer völlig anderen HTML-Seite gelangen, die auf eine geographisch weit entfernten Server gespeichert ist. Somit ist es möglich durch einfache Klicks vom einen Ende der Welt auf der anderen zu „navigieren" („surfen").

1.3 Grafik auf einer Webseite

Ausser den verschiedenen Textelementen kann auf einer HTML-Seite auch eine Grafik eingefügt werden. Die Grafikdatei sollte im Format „.gif" oder „.jpeg" vorliegen, das ziemlich mit allen Browser kompatibel ist. Dabei muss die Grafikdatei möglichst im gleichen Verzeichnis der entsprechenden Webseite gespeichert werden damit sie problemlos auf dem Bildschirm erscheint.
Wirklich Interessant wird es weiter mit verweissensitiven Grafiken. Diese sind Grafiken mit vordefinierten Bereiche, auf die man klicken kann. Wenn man mit dem Cursor über einen solchen Bereich zeigt, dann verwandelt sich der Cursor in einer kleinen Hand, was anzeigt, dass man hier klicken kann, um weiter Informationen zu erhalten. Wird ein solcher Klick ausgeführt wird man dann automatisch auf eine anderen Stelle innerhalb der Webseite verwiesen oder auf einer völlig neuen (vgl. Links).

1.4 Formulare

Mit Hilfe von Formulare kann man Informationen sammeln und vom Client-Rechner an einen Server versenden. In Formulare kann man die typische Elemente einbauen, die von graphischen Benutzeroberflächen her bekannt sind. Diese sind:

- Texteingabefelder,
- Eingabefelder für Passwort,
- Viereckige Auswahlkästchen, von denen mehrere ausgewählt werden können,
- Runde Auswahlkästchen, von denen jeweils nur eines ausgewählt werden kann,
- Schaltflächen für das Absenden oder Rücksetzen des Formulars,
- Aufklappbare Auswahlmenüs,
- Mehrzeilige Texteingabebereiche mit Scrollbars.

2 Dynamic HTML: Was ist das?

Die neueste Entwicklung auf dem Gebiet des Web-Publishing ist das Dynamische HTML (DHTML). „DHTML ist ein Oberbegriff für die neuren Technologien, um Web-Seiten ohne Rückgriff auf den Server dynamisch zu machen, und umfasst HTML, Style Sheets und Skriptsprachen (z.B. JavaScript).“[1] Was Style Sheets und Skriptsprachen sind werde ich später erklären. DHTML funktioniert nicht in allen Versionen der Microsoft- und Netscape-Browser. Erst ab der Version 4.0 des Microsoft Internet Explorer und ab der Version 4.0 des Netscape Communicator kann man mit DHTML arbeiten. Browser, die mit DHTML arbeiten, können verschiedene „dynamische“ Effekte automatisch oder interaktiv am Bildschirm auslösen. Solche können sein:

- Texte oder Bilder ändern beim Darüberfahren mit der Maus ihre Farbe oder ihre Grösse. Auch können plötzlich andere Texte oder Bilder erscheinen.
- Auf dem Bildschirm erscheinen schön fliessende Texte oder Animationen.
- Man hat die Möglichkeit verschiedene Seitenelemente mit der Maus zu verschieben.
- Es können verschieden Textschichten oder Menulisten übereinander liegen.
- Sogar komplexe, interaktive Spiele und Lernsituationen sind möglich.

Bei all diesen Effekten muss der Browser nicht auf den Server zurückgreifen und die Webseite neu laden. Das ganze läuft schon fliessend, und eben „dynamisch“ ab. Wenn man einige interessante Effekte selber erleben und ausprobieren möchte, so kann man diese unter „http://www.dansteinman.com/dynduo/“unter dem Titel „DHTML Demos“ aufrufen.

2.1 Grundlagen des DHTML

In der Tat ist DHTML keine eigentliche Erweiterung des HTML’s in Gestalt neuer HTML-Befehle. Es ist auch keine neue Programmiersprache. Vielmehr nützt DHTML neuere Technologien aus wie Style-Sheets und vor allem bestimmte neue Befehle in Scriptsprachen wie JavaScript. Die Basis jedoch Bleibt immer noch HTML. DHTML besteh also im wesentlichen aus drei Teile:

- **HTML** stellt die Grundbefehle (tags) mit Attribute zur Verfügung
- **Style Sheets** (Formatvorlagen) ordnen den HTML-tags Formateigenschaften zu
- **Scriptsprachen** können die HTML-tags, ihre Attribute und Formateigenschaften manipulieren und animieren.

Die „Textbearbeitung“ für Webseiten und die herkömmliche Textverarbeitung (z.B. mit MS Word) kann man folgendermassen gegenüberstellen:[2]

Web-Seiten	**Textverarbeitungssystem (z.B. Word)**
HTML	Textbearbeitung
Cascading Style Sheets	Formatvorlagen (z.B. Briefe oder Faxe)
Skripte (JavaScript oder VBScript)	„Makros“ (sind z.B. mit Visual Basic programmierbar)

[1] Frank Heston, Klaus Schoening: Dynamic HTML, S. 5
[2] Frank Heston, Klaus Schoening: Dynamic HTML, S. 13f

Dabei bedienen sich die Browsern eines Objektmodells. Dieser stellt eine hierarchisch angeordnete Menge von Objekten zur Verfügung, welche verschiedene Eigenschaften und Methoden haben. Über diese Objekte kann man mit Hilfe von Scriptsprachen auf die Einzelelemente einer Webseite zurückgreifen. Wie dieser Zugriff auf Webseitenelemente über Objekte funktioniert, erkläre ich im Kapitel 3.3.3 genauer.

Leider gibt es im DHTML auch einen Nachteil. Dieser ist dadurch entstanden, dass zwei Firmen, Microsoft und Netscape, unabhängig voneinander an der Entwicklung von DHTML beigetragen haben. Der Grundaufbau ist zwar immer gleich, jedoch lassen sich grob zwei Versionen von DHTML unterscheiden: eine Microsoft- und eine Netscape-Version.
Dieser Unterschied beruht darauf, dass die Objektmodellstruktur des Microsoft-Browser und diejenige des Netscape-Browser unterschiedlich aufgebaut ist. Darum muss man beim Programmieren von HTML immer spezifizieren ob man mit Microsoft- oder Netscape-DHTML arbeitet.
Will man hingegen eine DHTML-Seite programmieren die auf beiden Browsern läuft, so gibt es hierfür eine Cross-Browser-Lösung, die sowohl von Netscape als auch von Microsoft verstanden wird. Um die Unterschiede zwischen Microsoft- und Netscape-DHTML zu erläutern, zeigen ich später im Kapitel 4 zwei Beispiele.

Wie bereits erwähnt ist DHTML aus drei Grundelemente aufgebaut: HTML, Formatvorlagen (Cascading Style Sheets) und Scriptsprache (JavaScript). HTML habe ich schon im ersten Kapitel kurz vorgestellt. Im folgenden werde ich auf die zwei weiteren Grundlagen des DHTML näher eingehen.

2.2 Formatvorlagen (Style Sheets)

Formatvorlagen oder Style-Sheets sind eine unmittelbare Ergänzung zum HTML. Mit HTML kann man Webseitenelemente definieren, für die bestimmte Standardeigenschaften gelten. Mit Formatvorlagen ist es möglich diese Standardeigenschaften zu ändern. Beispielsweise kann man mit Style-Sheets eine Überschrift 1. Ordnung definieren, die eine andere Schriftgrösse, -art oder –formatierung (fett oder kursiv) hat als üblich. Ohne die Formatvorlage würden die Überschrift in ihrer Standardausprägung immer gleich erscheinen.
Weiter können wir auch beliebige Bereiche einer HTML-Datei mit einer eigenen Hintergrundfarbe, einem eigenen Hintergrundsbild oder mit verschiedenen Rahmen versehen. Auch können wir einzelne Webseitenelemente (Grafik, Textabsatz, Tabelle) pixelgenau im Browserfenster platzieren.
Es gibt mehrere Sprachen zum Definieren von Style-Sheets. Die bekannteste ist die CSS-Sprache. CSS steht für "Cascading Style Sheets" und wurde vom World Wide Web Consortium eingeführt. Mit Cascading Style Sheets (CSS1) kann man Formatvorlagen lokal für einen tag oder global für eine Gruppe von tags definieren. Die so definierten Formatvorlagen sind wie eine Kaskade ineinander verschachtelt, was die Bedeutung von „Cascading“ verdeutlicht.
Es gibt also verschiede Arten, wie man HTML-tags Formatvorlagen zuordnen kann:

- **Inline-Formate** werden einfach im betreffenden tag eingeschrieben. Diese Eigenschaften beziehen sich nur auf einen bestimmten tag (lokale Definition). Zum Beispiel:

```
<p style=“fontsize:14pt“>Dies ist ein Absatz</p>
```

Diese Zeile beschreibt einen beliebigen Absatz einer HTML-Seite mit einer Schriftgrösse von 14 Punkte.

- **Eingebettete Formate** sind Formatvorlagen, die getrennt zwischen head- und body-Teil einer HTML-Seite platziert werden und global für die ganze HTML-Seite gelten. Eingebettete Formate stehen in einem <style> </style>-Block.

```
<html>
<head>
<title>Titel der Webseite</title>
</head>
<style>
... Hier kann man spezielle Eigenschaften, z. B. von
Überschriften oder Absätze, definieren ...
</style>
<body>
... Inhalt der Webseite ...
</body>
</html>
```

- **Verknüpfte Formatdefinitionen** stehen in einer eigenen Datei. Diese externe Datei kann zum Beispiel die Ausprägung „.css“ haben. Mit einer solchen css-Datei kann man Formatvorlagen definieren, die für eine ganze Gruppe von HTML-Seiten gelten sollen. Dies erspart die mühsame lokale oder globale Definition innerhalb der einzelnen HTML-Seiten. Um eine Webseite mit einer Formatvorlagen-Datei zu verknüpfen, muss man die Webseite mit einem <link>-tag versehen, der einen entsprechenden Verweis auf die css-Datei beinhaltet. Zum Beispiel:

```
<link rel=stylesheet href=“mystyle.css“ type=“text/css“>
```

rel=stylesheet verweist auf einer Formatvorlage, href=“mystyle.css“ beinhaltet den Dateinamen der Formatvorlagendatei und type=“text/css“ gibt die Datenart an (in diesem Fall css = Cascading Style Sheets).

- **Importieren des Style Sheets** in die Datei, ist eine weitere Möglichkeit um eine externe Datei zu nutzen. Dabei wird die Datei automatisch in die Webseite eingefügt. Zum Beispiel:

```
<style type="text/css">
@import url (style1.css)
</style>
```

In einer HTML-Datei können auch mehrere Arten von Formatvorlagendefinitionen vorkommen. In diesem Fall haben dann individuelle, lokale Formate vor allgemeinen, globalen Formatdefinitionen den Vortritt. Somit kann man eine ganze Gruppe von HTML-Dateien mit den gleichen Formatdefinitionen versehen und einzelne Dateien oder sogar Seitenelemente mit Eingebettete oder Inline-Formate abändern. Diese Verschachtelung der Formatdefinitionen in einer Art „Kaskade“ erklärt die Bedeutung von „Cascading“ in „Cascading Style Sheets“.

2.3 JavaScript

JavaScript ist kein direkter Bestandteil von HTML, sondern eine eigene Programmiersprache. Diese Sprache verfolgt zwar einige Grundideen der bekannten Programmiersprache Java, es wurde jedoch zu dem Zweck geschaffen, HTML-Programmierern ein Werkzeug zu geben, um Webseiten zu optimieren und dynamisch zu gestalten. Mit Hilfe von Scriptsprachen werden also kleine Anwendungen und Methoden definiert, die direkt auf dem Client-Computer ausgeführt werden können ohne auf den Server zurückgreifen zu müssen. Die Programmcodes werden also vom jeweiligen Browser, Microsoft oder Netscape, decodiert und ausgeführt ohne die Webseite neu zu laden.
JavaScript ist eine Scriptsprache die von Netscape entwickelt und lizenziert wurde. Deshalb ist die neuen Version des Netscape-Browser dem Microsoft-Browser jeweils einen Schritt im voraus. Netscape interpretiert seit Version 4.0 den JavaScript-Standard 1.2. Einige Befehle dieser Sprachversion werden auch vom Microsoft Internet Explorer 4.0 verstanden. Der Microsoft Internet Explorer 4.x interpretiert daneben aber auch die Microsoft-eigene Sprachvariante Jscript und zusätzlich die eigene Skriptsprache VBScript (Visual Basic Script).
Beide Scriptsprachen orientieren sich jedoch am Standard für Internet-Scriptsprachen, an **ECMA-262**. Das ECMA-Kommittee, dem verschiedene Software-Hersteller angehören, bemüht sich, auch im Bereich der Scriptsprachen einen allgemeinen Standard zu definieren, so wie es bei HTML oder CSS Style Sheets der Fall war.

Die Problematik von DHTML besteht darin, dass JavaScript über die Objektmodelle der Browser auf die Webseitenelemente zugreift. Da die Objektmodelle der beiden Browser unterschiedlich aufgebaut sind, ist auch der Zugriff auf den jeweiligen Objekten unterschiedlich zu codieren. Die Herausforderung besteht nun darin mit Hilfe einer Cross-Browser-Lösung die unterschiedlichen Arten des Zugriffs zu verallgemeinern. Im Kapitel 4 führe ich ein solches Beispiel auf.

Im Gegensatz zu HTML und CSS1 ist JavaScript „case-sensitive“, die Gross-/Kleinschreibung spielt also beim Programmieren eine wichtige Rolle.
In einer Programmiersprache wie JavaScript gibt es für Anfänger viele verwirrende Elemente: Sonderzeichen, Variablen, Wenn-Dann-Anweisungen, Schleifen, Funktionen, Methoden, Parameter, Objekte, Eigenschaften und anderes mehr. Um mit diesen Elementen richtig umgehen zu können, ist ein langwieriger Lernprozess und viel Übung nötig. JavaScript ist dazu allerdings gut geeignet, weil es eine relativ einfache Sprache ist, bei der viele Aufgabenbereiche einer vollständigen Programmiersprache fehlen, zum Beispiel Dinge wie Arbeitsspeicherverwaltung oder Dateioperationen. Außerdem ist JavaScript gut durchschaubar, weil sich seine Programmrcodes stets auf die angezeigte oder anzuzeigende Webseite beziehen.

2.3.1 Funktionen

„Eine Funktion ist ein Anweisungsblock. Ein Anweisungsblock besteht aus zwei oder mehreren Anweisungen (Befehle), die innerhalb einer übergeordneten Anweisung oder innerhalb einer Funktion stehen. So können Anweisungsblöcke beispielsweise innerhalb einer bedingten Anweisung (Wenn-Dann-Anweisung) oder innerhalb einer Schleife stehen. Auch alle Anweisungen, die innerhalb einer selbst definierten Funktion stehen, bilden einen Anweisungsblock.“[1] Eine einfache Funktion wird so definiert:

[1] http://www.teamone.de/selfhtml/teba.htm#a2

```
function zeigeText() {
alert (“Hallo!“);
}
```

Mit Hilfe von Funktionen kann man eigene, in sich abgeschlossene JavaScript-Prozeduren programmieren, die dann beim Aufruf der Funktion ausgeführt werden. Dabei kann dieser Aufruf auf verschiedenen Arten erfolgen, zum Beispiel beim Klicken mit der Maus auf einen Button. Wenn die Prozedur hingegen nicht innerhalb einer Funktion steht, so wird diese direkt beim Einlesen der Webseite ausgeführt.
Funktionen können irgendwo auf dem HTML-Dokument definiert werden. Am besten platziert man sie jedoch im oder unter dem Head-Teil. Dies ist besser, weil eine Webseite stückweise geladen wird. Wenn also ein Funktion am Ende einer HTML-Seite steht, kann es vorkommen, dass zum Beispiel Buttons und Textfelder am Bildschirm erscheinen bevor die zugehörige Funktion bereitsteht. Klickt der Anwender jetzt auf den Knopf erscheint eine Fehlermeldung.
Der grundlegende Aufbau einer Webseite mit DHTML sieht darum folgendermassen aus:

2. Style Sheets	<html> <head> <title>Titel der Webseite</title> </head> <style> ...Hier definieren wir unsere Style Sheets... </style>
3. JavaScript-Funktionen	<script language=“JavaScript“> ...Hier kommen unsere JavaScript-Funktionen... </script>
1. HTML	<body> ...Hier bauen wir unsere Webseite mit HTML-tags auf... </body> </html>

Die JavaScript-Funktionen stehen in einem <script> </script> Block. Um zwischen den verschiedenen Skriptsprachen zu unterscheiden, hat man das Attribut „language=...“ eingeführt. Im Falle von JavaScript schreiben wir: language=“JavaScript“. Steht der Code hingegen in VBScript (Visual Basic Script) schreiben wir: language=“VBScript“.

Funktionen können auch in einer separaten JavaScript-Datei definiert werden. Das ist sehr nützlich, wenn wir gleiche JavaScript-Funktionen in mehreren HTML-Dateien verwenden wollen. Auf der HTML-Seite muss dann nur noch einen Verweis darauf gesetzt werden:

```
<script language="JavaScript" src="funktion.js"
type="text/javascript">
</script>
```

Wie bei den externen Formatvorlagen muss auch hier die Quelldatei angegeben werden (src=source=Quelle), in diesem Fall handelt es sich um eine js-Datei, und die Datenart (MIME-Type) ist JavaScript.

2.3.2 Event-Handler

Funktionen können nicht nur von anderen Funktionen aufgerufen werden sondern auch mit Hilfe von Event-Handler. Event-Handler (Ereignis-Behandler) sind ein wichtiges Bindeglied zwischen HTML und JavaScript. Sie werden ganz einfach in Form von Attributen in HTML-tags geschrieben. Beispielsweise:

```
<img src="grafik.jpg" onClick="zeigeText()">
```

In diesem Beispiel ist „onClick" ein Event-Handler, er „behandelt" sozusagen Ereignisse, indem er eine JavaScript-Funktion aufruft. In diesem Fall wird die Funktion „zeigeText" aufgerufen sobald man auf das Image (grafik.jpg) klickt. Event-Handler beginnen immer mit „on" und danach wird der Name des entsprechenden Ereignisses direkt angehängt (on + Click = onClick). Insgesamt gibt es über 20 Event-Handler, den Sinn der meisten kann man, mit ein bisschen Englischkenntnisse, aus deren Namen ablesen. Zum Beispiel: onMouseOver, wenn man den Cursor über ein Seitenelement verschiebt; onMouseMove, wenn man die Maus mit gedrückter Maustaste bewegt; onAbort, wenn man im Browser auf den Stop-Button drückt.

2.3.3 Objekte

Das Objekt-Konzept geht davon aus, dass jedes Browserelement ein Objekt darstellt. Diese Objekte sind von einem bestimmten Typ (Klasse). Objekte, die von der gleichen Klasse stammen haben gleiche Eigenschaften und Methoden, jedoch mit unterschiedlichen Werte. Bei JavaScript gibt es vier Arten von Objekten:

- **HTML-Objekte**
  HTML-Objekte umfassen praktisch alle tags einer Webseite, die mit JavaScript manipuliert werden können. Zum Beispiel: form, button, links, images.

- **Browser-Objekte**
  Im wesentlichen versteht man darunter die Objekte window, document, history, location, navigator. HTML- als auch Browser-Objekte sind konkrete Objekte, die beim Laden der Seite automatisch angelegt werden und können einfach benützt werden.

- **Vordefinierte Objekte**
  Bei den vordefinierten Objekten handelt es sich um Objekttypen, die sozusagen als „Kopievorlage" für die Objekte dienen. Aus einem Objekttyp können wir beliebig viele Objekte erzeugen, die alle für diesen Objekttyp definierte Eigenschaften und Methoden besitzen. Beispiele sind: String, Array, Date, Math, Image.

- **Selbstdefinierte Objekte**

Auch hier handelt es sich um Objekttypen, mit dem Unterschied, dass wir hier die Eigenschaften und Methoden selbst definieren müssen.

Die Objekte sind hierarchisch geordnet. Will man nun auf ein bestimmtes Element einer Webseite zugreifen, muss man ihn über die Objekthierarchie der Seite lokalisieren. Dies geschieht mit dem Punktoperator, zum Beispiel:

```
window.document.form1.text1
```

Dabei steht „window“ für das Browserfenster, „document“ für deren Inhalt, „form1“ verweist weiter auf das Formular mit dem Namen „form1“ und schliesslich „text1“ auf deren Texteingabefeld „text1“. Wenn wir nun den Textinhalt von „text1“ ändern wolle müssen wir nur schreiben:

```
window.document.form1.text1.value=”neuer Text”
```

Die Änderung der Eigenschaften von Objekten kann man auch eleganter formulieren, indem man eine Funktion schreibt, die diese Tätigkeit ausführt. Man kann zum Beispiel eine Methode „changeText“ einführen, um den Inhalt des Texteingabefeldes „text1“ zu ändern. Also:

```
window.document.form1.tex1.changeText(“neuer Text”)
```

2.3.4 Fenster und Frames

Mit HTML und JavaScript kann man auch Webseiten erstellen, die aus mehreren Unterfenstern bestehen (Frames). Dabei stellt jedes Unterfenster eine unabhängige Webseite dar, das heisst dass jedes Frame als eigenständige Webseite geladen werden kann. Zum Beispiel kann eine Webseite aus zwei Unterfenstern bestehen. Das linke Unterfenster könnte ein Menüleiste mit verschiedenen Buttons darstellen, und das rechte könnte als Anzeigefläche dienen. Beim Klicken auf ein Button in der Menüleiste würde dann in der Anzeigefläche die entsprechenden Webseite geladen.
Frames sind auch in HTML möglich. Hier sieht man also schon, dass man HTML und DHTML nicht klar abgrenzen kann. Allerdings sind Frames in HTML nicht besonders praktisch, denn jedes Mal wenn eine neue Webseite im Unterfenster aufgerufen wird, muss diese neu geladen werden. In DHTML muss ein Unterfenster hingegen nicht immer neu geladen werden, sonder es kann einen einfachen Schichtenwechsel stattfinden. Denn in DHTML können ganze Schichten übereinander gelegt werden, wobei nur die oberste angezeigt wird (vgl. Schichteneffekte im Kap. 3.4).

3 Wie realisiere ich DHTML

Mit DHTML kann man nun eine Webseite dynamisch ändern. Es gibt unendlich viele neue Effekte, die man mit DHTML darstellen kann. Dabei besteht ein Problem. Je nachdem, wie die DHTML-Seiten geschrieben sind, können sie entweder nur im Microsoft Internet Explorer ab 4.0 (IE 4.0), nur im Netscape Communicator ab 4.0 (NC 4.0) oder in beiden Browsern dargestellt werden. Versionen, die in beiden Browsern laufen nennt man Cross-Browser-Lösungen. Wir sind zwar bereits schon bei der Version 5.5 des MS Internet Explorer und der Version 6.0 des Netscape Communicator (stand Mai 2001) angelangt, ich beschränke mich aber im Folgenden auf den Versionen 4.0, da diese die ersten sind, die DHTML interpretieren können. Die weiteren Versionen interpretieren DHTML natürlich auch problemlos.
Als nächstes möchte ich eine DHTML-Seite einmal mit IE 4.0 und einmal mit NC 4.0 erzeugen und danach deren Cross-Browser-Lösung zeigen. Schliesslich werde ich weitere Effekte erwähnen, die mit DHTML möglich sind. Diese Effekte werde ich nur kurz beschreiben, ich werde aber nicht auf deren Programmierung eingehen.

Zwischen HTML und DHTML gibt es keine klare Unterscheidung. DHTML ist mehr „eine logische Weiterentwicklung, dass man über das Objektmodell auf einzelne Elemente einer Webseite zugreifen kann.“[1] Denn bei IE 3.0 und NC 3.0 konnte man bereits auf eine beschränkte Menge von tags zugreifen, für die ein Objekt definiert war. Vorallem war das bei Formularelemente (Buttons und Texteingabefelder) der Fall, im Ganzen jedoch bei ca. 15 Objekttypen.

Bei den Versionen 4.0 ist nun für jeden möglichen tag ein Objekt definiert, von der kleinsten Überschrift bis zur horizontalen Linie. Man hat somit Zugriff nicht nur auf allen existierenden HTML-tags sondern auch auf allen HTML-Attribute, die es gibt. Bei der Versionen 4.0 stehen gesamthaft ungefähr 80 Objekte zur Verfügung.

Man kann sich jetzt einfach vorstellen, warum die Objektmodelle der beiden Browsern so unterschiedlich aufgebaut sind. In der ursprünglichen 3.0-Version waren die Objektmodelle von IE und NC im Grundaufbau sehr ähnlich. Durch die Erweiterung auf die 4.0-Versionen haben sich die Objektmodelle der beiden Browsern auf unterschiedliche Weise weiterentwickelt.

Ich möchte nun am Beispiel der Schichtendarstellung mit DHTML diese Problematik erläutern. Eine Schicht wird in den beiden Browsern zwar mit denselben tag (-tag) aufgerufen, sie wird aber in unterschiedlicher Weise bearbeitet.

3.1 DHTML in Microsoft[2]

Es soll eine Webseite dargestellt werden bei der vorerst nur zwei Buttons erscheinen. Beim Drücken auf das eine Button soll dann darunter einen kurzen Text erscheinen, beim Drücken auf den anderen soll dieser Text wieder verschwinden. Wie geht man da im Microsoft IE 4.0 vor.
Zuerst muss man den anzuzeigenden Text in Form einer Schicht (Textbschnitt) definieren. Eine Schicht fügen wir mit dem div-tag ein, div steht für „division“:

[1] Frank Heston, Klaus Schoening: Dynamic HTML, S. 142
[2] Alle Codes im folgenden Text sind aus Frank Heston, Klaus Schoening: Dynamic HTML, S. 147f

```
<div id=“ebene1“ style=”color: red; visibility:hidden;”>
Hier kommt der vorerst unsichtbare Text
</div>
```

Mit dem Attribut id=”ebene1” kann man der div-Schicht den Namen „ebene1“ zuordnen und macht sie automatisch zu einem Objekt. Die Farbe der Schicht soll weiter rot sein und am Anfang versteckt (=hidden) bleiben. Diese beiden Eigenschaften werden mit dem Attribut „style=“ festgehalten. Zwischen und kommt dann der Text, der beim Clicken auf den Button erscheinen soll.

Jetzt müssen die Funktionen definiert werden, welche die Schicht erscheinen und wieder verschwinden lassen. Die beiden Funktionen nennen wir „zeigeSchicht()“ und „verbergeSchicht()“. Eine Schicht ist bei Microsoft IE als all-Objekt definiert. All-Objekte sind wiederum Teil eines „dokument“-Objekt, welcher seinerseits unter das „window“-Objekt steht. Um die Schicht „ebene1“ anzusprechen schreiben wir also:

```
window.document.all[„ebene1“]
```

oder verkürzt:

```
window.document.ebene1
```

bzw. ganz kurz:

```
ebene1
```

Um die Schicht bei Knopfdruck anzuzeigen oder zu verbergen, müssen wir die Eigenschaft der Schicht von unsichtbar auf sichtbar setzen. Die Eigenschaften einer Schicht stehen bekanntlich unter dem Attribut „style=“. Das style-Attribut oder style-Objekt ist ein Unterobjekt der div-Schicht. Die Eigenschaft einer Schicht können wir also so ansprechen:

```
eben1.style
```

Um die Eigenschaft auf sichtbar zu setzen, müssen wir den Attribut von visibility:hidden auf visibility:visible setzen. Die Funktion „zeigeSchicht()“ für das Anzeigen lautet also:

```
function zeigeSchicht(){
ebene1.style.visibiliti=“visible“;
}
```

Die Funktion „verbergeSchicht()“ für das Verbergen hingegen:

```
function verbergeSchicht(){
ebene1.style.visibiliti=“hidden“;
}
```

3.2 DHTML in Netscape[1]

Dieselbe Webseite soll nun im Netscape Communicator 4.0 programmiert werden.

Die Anfangsdefinition der Schicht ist in den beiden Browsern nicht genau gleich. In der „style"-Definition muss noch die absolute Positionierung angegeben werden, weil sonst die Seite nicht richtig funktioniert. Also:

```
<div id="ebene1" style="position:absolute; color:red;
visibility:hidden">
```

Das Objektmodell des NC 4.0 kennt das all-Objekt nicht. Eine Schicht wird unter den Namen „layer" als Objekt angesprochen. Div-Schichten werden also einfach als layer-Objekte abgelegt. Überspitzt ausgedruckt kann man sagen: „div-Schichten beim NC 4.0 sehen aus wie div-Schichten, werden aber behandelt, als wären sie layer-Schichten."[2]
Das Objektmodell des NC 4.0 ist also folgendermassen aufgebaut: Zuerst kommt wieder das window-Objekt, darunter das document-Objekt und dann das layer-Objekt, zu dem auch Schichten gehören. Um die div-Schicht anzusprechen, schreiben wir also:

```
window.document.layer[„ebene1"]
```

oder verkürzt:

```
document.ebene1
```

Beim NC 4.0 gibt es auch kein style-Objekt, d. h. man kann die visibility-Eigenschaft einfach an die Schicht „ebene1" anhängen. Um die Schicht anzuzeigen muss jetzt aber den Wert „show" eingegeben werden, um sie zu verbergen den Wert „hide".
Für die Funktion „zeigeSchicht()" ergibt sich folgender Code:

```
document.ebene1.visibility="show"
```

Und für die Funktion „verbergeSchicht()" folgender:

```
document.ebene1.visibility="hide"
```

3.3 Cross-Browser-Lösung[3]

Wie können wir nun die gleiche HTML-Seite schreiben, damit sie sowohl vom IE 4.0 als auch vom NC 4.0 dargestellt werden kann?
Wie wir gesehen haben sind die Funktionen „zeigeSchicht" und „verbergeSchicht" im IE 4.0 und im NC 4.0 unterschiedlich aufgebaut. Um zwischen einer Funktion im IE 4.0 und einer Funktion im NC 4.0 unterscheiden zu können, brauchen wir ein Unterscheidungskriterium.

[1] Alle Codes im folgenden Text sind aus Frank Heston, Klaus Schoening: Dynamic HTML, S. 152f
[2] Frank Heston, Klaus Schoening: Dynamic HTML, S. 151
[3] Alle Codes im folgenden Text sind aus Frank Heston, Klaus Schoening: Dynamic HTML, S. 155f

In der Cross-Browser-Lösung werden sozusagen beide Funktionen aufgeschrieben, der Browser arbeitet dann mit derjenigen weiter, die er versteht.
In DHTML schreibt man als Unterscheidungskriterium üblicherweise:

```
If (document.layers)
      ... hier folgt der Netscape-Code...
else
      ... hier folgt der Microsoft-Code...
```

Hier wird einfach abgefragt, ob das Objektmodell das layer-Objekt enthält. Im „if"- Teil, falls das layer-Objekt bekannt ist, kommt der Netscape-Code, im „else"- Teil, falls das layer-Objekt unbekannt ist, kommt der Microsoft-Code.
Wie sehen die Funktionen „zeigeSchicht()" und „verbergeSchicht()" auf Cross-Browser aus? Die beiden Funktionen müssen diesmal mit **Übergabewerte** arbeiten. Dies wird beim Event-Handler „onClick" bei der Definition des Buttons festgelgt. Hier lautet der Übergabewert „ebene1":

```
onClick="zeigeSchicht('ebene1')"
```

bzw.

```
onClick="verbergeSchicht('ebene1')"
```

Der Übergabewert steht dabei zwischen den Klammern des Funktionsnamen, die bis jetzt immer leer geblieben sind. In diesem Fall wird einen Code verwendet der zwischen Hochkommata steht. Innerhalb von diesen Code erscheint ein weiterer Code der wiederum Hochkommata erfordert. Um zwischen den beiden Codes zu unterscheiden schreiben wir einmal doppelte Hochkommata (") und einmal einfache Hochkommata (').
Die Funktion „zeigeSchicht()" nimmt den Übergabewert entgegen und arbeitet mit diesen weiter. Dabei muss sie zwischen Microsoft- und Netscape-HTML unterscheiden und entsprechend die visibility-Eigenschaft entweder auf „show" bzw. „visible" setzen:

```
function zeigeSchicht (x){
     if (document.layers)
     document.layers[x].visibility="show";
     else
     document.all[x].style.visibility="visible";
     }
```

Die Funktion „verbergeSchicht()" setzt die visibility-Eigenschaft entsprechend auf „hide" bzw. „hidden":

```
function hideLayer(x){
     if (document.layers)
     document.layers[x].visibility="hide";
     else
     document.all[x].style.visibility="hidden";
     }
```

3.4 Weitere Effekte mit DHTML

Im folgenden möchte ich eine Reihe weiterer Effekte aufzählen, die mit DHTML möglich sind. Wie schon gesagt gibt es unzählig viele neue Effekte, die man mit DHTML darstellen kann. Auf deren umständlichen Programmierung und Unterscheidung zwischen Microsoft-, Netscape- und Cross-Browser-HTML werde ich nicht mehr eingehen. Auch gibt es Effekte die entweder nur ab IE 4.0 oder ab NC 4.0 möglich sind. Als klassische Beispiele gelten:

- **Textformatierung und Textinhalt dynamisch ändern**
  Man kann eine Überschrift oder einen Textabschnitt so programmieren, dass sie „dynamisch" ändern. Dynamisch heisst dabei, dass sobald man mit dem Cursor draufklickt oder drüberfährt ihre Grösse, Farbe, Schriftart, Ränderabstand, Position oder Inhalt ändert. Dynamisch kann auch bedeuten, dass solche Änderungen automatisch in regelmässigen Zeitintervallen stattfinden.

- **Schichteneffekte**
  Schichten oder Ebenen können beim Laden einer Webseite vorerst unsichtbar sein. Sobald man mit dem Cursor über einen bestimmte Grafik oder Button auf der Webseite drübergeht oder klickt erscheinen diese plötzlich.
  Man kann auch Textschichten durchsichtig aufeinander legen, so dass man die verschiedenen Textschichten übereinander liegen sieht. Oder hingegen undurchsichtig aufeinander legen, so dass man jeweils nur die oberste Schicht sieht. Textschichten lassen sich auch problemlos bewegen. Dadurch kann man besondere Effekte erzielen: Text ein-/ausblenden, Textabschnitte versetzen oder in der Ausrichtung ändern.

- **Grafikinhalt dynamisch ändern**
  Nicht nur Texte und Überschriften können dynamisch ändern sonder auch Grafiken. Dies ist vor allem nützlich, wenn man eine Web-Dias-Show durchführen möchte mit Kommentar zu den Bildern.

- **Elementposition dynamisch ändern**
  Mit DHTML kann man Grafiken darstellen, die man mit der Maus verschieben kann.

- **Überblendeffekte**
  Mit Überblendeffekte können ganze Seiten übergeblendet oder auch bestimmte Teile einer Seite ein- oder ausgeblendet werden. Ein Titelblatt einer Webseite könnte dann beim Aufruf von rechts nach links langsam angezeigt werden, so als ob ein Buch aufgeschlagen würden.

- **Roll- und Pull-Down-Menüs**
  Will man eine Webseite mit einer Menüleiste mit vielen Untermenüs darstellen, müssen diese nicht alle zur gleichen Zeit sichtbar sein, sondern wie von normalen Desktop-Anwendungen her bekannt, erst bei Bedarf aufklappen. Solche Untermenüs können langsam ausgerollt werden oder einfach erscheinen und wieder verschwinden.

- **Laufschrift**
  Für Werbezwecke oder um die Aufmerksamkeit zu erwecken, können Laufschriften unterschiedlicher Grösse und Geschwindigkeit programmiert werden.

4 Zusammenfassung und Ausblick

Ich hoffe, dass diese Arbeit einen guten Überblick über die Möglichkeiten des DHTML im Vergleich zum herkömmlichen HTML geben konnte. Am Ende sollte jeder sagen können, dass DHTML ganz einfach sei. Denn es gehört nichts anderes dazu als

- das herkömmliche **HTML**, mit deren tags ich die Webseite optisch aufbaue,
- **Style Sheets**, mit denen ich den tags Formatvorlagen zuordne und
- **JavaScript**, womit ich tags, Attribut und Formateigenschaften dynamisch manipulieren kann

DHTML ist also keine völlig neue Programmiersprache, sondern lediglich eine Erweiterung des HTML um den erwähnten Komponenten. Mit DHTML will man sich lediglich an die neuen Entwicklungen des Web-Publishing anpassen, nämlich auf das dynamische Ändern von Webseiten ohne auf den Server zurückgreifen zu müssen. Die alten Zeiten mit einfachen Webseiten, die jedermann mit ein bisschen HTML-Kenntnissen selber entwerfen konnte, sind vorbei. Heute trifft man zum Teil auf hochdynamische Webseiten mit bewegte Bilder, grandiose Hollywood-Effekte und sogar ganze Computerspiele, die jedoch nicht nur mit DHTML programmierbar sind. Denn auf dem Markt sind auch andere Programme vorhanden wie „Macromedia Flash“ oder Java mit denen sich genauso schöne „dynamische“ Webseiten programmieren lassen. Dadurch muss man nichts anderes tun als den Browser mit Flashplayer oder Java Virtual Mashine aufstocken und schon läuft das Ganze. Allerdings werden dadurch Webseiten immer umfangreicher, haben längere Ladezeit und können nicht mehr von jedermann kreiert werden, der keine grosse Computerkenntnisse hat.
Das einzige Problem mit DHTML ist, dass es zur Zeit hauptsächlich zwei verschiedene Versionen gibt, eine von Microsoft Internet Explorer (ab 4.0) und eine von Netscape Communicator (ab 4.0). Somit ist man gezwungen entweder eine Sprache auszuwählen oder eine aufwendige Cross-Browser-Lösung zu umschreiben. Deswegen ziehen manche Programmierer vor eine Webseite mit Macromedia Flash oder Java zu erweitern um sie dynamisch zu gestalten. Die Grundstruktur bleibt jedoch in HTML. Wenn man also keine Einigung zwischen Microsoft und Netscape findet, ist die Zukunftsfähigkeit von DHTML von anderen Web-Publishing-Produkten bedroht.

Mit meiner Arbeit will ich wie gesagt nur einen groben Überblick über die Möglichkeiten des DHTML verschaffen. Falls sich jemanden im Thema JavaScript vertiefen will oder falls jemanden mit der Syntax oder das Objektmodell Schwierigkeiten hat, empfehle ich ein gutes Buch über JavaScript. Auch im Internet hat es unzählig viele Webseiten zum Thema DHTML, wo man auch die dynamische Effekte direkt am Bildschirm anschauen kann.
Natürlich muss man DHTML nicht im Detail lernen, um eine schöne Webseite zu erstellen. Wenn man im World Wide Web auf eine DHTML-Seite stosst, die einem gefällt, so kann man ganz einfach deren Quelltext anzeigen lassen und nachschauen, wie das zu realisieren ist. Man kann auch ganz bestimmte Webseitenelemente aus einem Quellcode kopieren und in der eigenen Webseite einfügen. Dazu muss man nicht unbedingt alles genau programmieren können, sonder es genügt lediglich, wenn man den Code versteht und entsprechend ändern und anpassen kann.

Literaturverzeichnis

- Frank Heston, Klaus Schoening: Dynamic HTML, Publics MCD Verlag, München 1999

- David Flanagan: JavaScript – kurz & gut (Deutsche Ausgabe), O' Reilly Verlag, Köln 1998

Internetseiten:

- Stefan Münz: SELFHTML – HTML-Dateien selbst erstellen
  http://www.teamone.de/selfhtml/

- Dan Steinman: The Dynamic Duo – Cross-Browser Dynamic HTML
  http://www.dansteinman.com/dynduo/

- Christoph Bergmann, Hannes Gamperl: Milch & Zucker – Die deutsche Dynamic HTML Seite
  http://dhtml.seite.net/

- Hubert Partl: HTML-Einführung
  http://www.boku.ac.at/htmleinf/hein.html

- Bernd Buss, Netgr@phics – DHTML – Die Zukunft des WebDesign
  http://www.net-graphics.de/Dhtml/dhtml1.htm