

Webseiten mit dem Document Object Modell

Dynamic HTML

DHTML – „Dynamisches“ HTML ist die **Erweiterung vom normalen statischen HTML** durch ein Objektmodell. Lesen Sie, wie Sie das DHTML-Objektmodell verwenden, um interaktive Seiten zu gestalten.

THOMAS WÖLFER

Das DHTML-Objektmodell liegt zwar dem W3C vor, und neuere Browser-Versionen unterstützen es auch im Großen und Ganzen, dennoch sind die Unterschiede der Implementierung beim NetScape und beim Microsoft-Browser teilweise dramatisch. Daher wurden die folgenden Beispiele nur mit neueren Microsoft-Browsern getestet. Mit Hilfe von „außen“ sollte es aber möglich sein, „Cross-Browser“-Scripts zu schreiben, die das DOM nutzen. Eine gute Informationsquelle für solche Vorgänge ist die „Inside HTML“-Seite unter:

<http://www.siteexperts.com/>

■ Das DHTML-Objektmodell

Das DHTML-Objektmodell erlaubt den Zugriff auf alle Aspekte des im Browser geladenen Dokuments. Inner-

halb des Objektmodells ist das Window-Objekt das in der Hierarchie am höchsten stehende. Unter anderem beinhaltet dieses Objekt das „document“-Objekt, das den eigentlichen Inhalt des Dokuments darstellt. Alle Zugriffe auf das Dokumenten-Objekt erfolgen über das Window-Objekt.

Das Window-Objekt bietet außerdem Informationen über die URL des aktuell im Browser angezeigten Dokuments sowie über URLs, die der Client (der Browser) bereits besucht hat.

Bei jeder Veränderung eines Aspekts des Dokuments durch das Objektmodell – zum Beispiel durch die Vergrößerung eines Teils des Textes – berechnet das Dokument sein Layout neu und zeigt es unter Berücksichtigung der neuen Attribute im Browser an.

Der Zugriff auf das Objektmodell erfolgt auf eine von drei möglichen Arten:

- **Mit Scripts:** Scripts sind entweder direkt in der Seite enthalten oder befinden sich auf einer anderen Seite, die dann von der aktuellen Seite referenziert wird. Scripts können sowohl den Inhalt als auch den Stil des aktuellen Dokuments beeinflussen. Prinzipiell ist das Objektmodell von der verwendeten Script-Sprache unabhängig; meist wird aber entweder ein Ja-

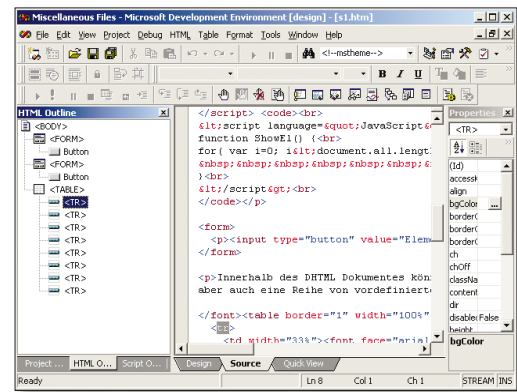
vaScript Dialekt oder VBScript verwendet.

- **Mit Applets oder Controls:** Sowohl Applets als auch Controls müssen in der Seite enthalten sein. Ein Control kann praktisch in jeder Sprache implementiert werden, zum Beispiel in Visual Basic oder in C++. Dabei ist allerdings zu beachten, dass ein mit C++ für Win32 programmiertes Control logischerweise nicht auf anderen Plattformen – zum Beispiel unter Linux – ausgeführt werden kann.

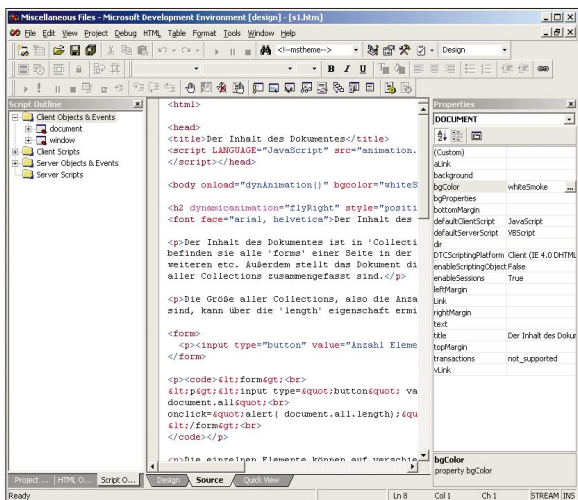
- **Durch andere Hosts:** Prinzipiell ist es möglich einen Host zu implementieren, der sich außerhalb des Browsers befindet oder eine Browser-Erweiterung darstellt.

Scripts sind die Methode, mit der am häufigsten auf das Objektmodell zugegriffen wird. Alle Beispiele in diesem Beitrag verwenden JScript als Script-Sprache. Den Quell-Code zu diesen Beispielen finden auf der Heft-CD.

Eine praktische Eigenschaft von DHTML ist, die Scripts außerhalb des Dokuments (bzw. der Seite) in einer externen Script-Bibliothek zu lagern. Auf diese Weise ist es zum einen sehr einfach möglich Scripts wiederzuverwenden.



MIT DER „DOCUMENT OUTLINE“ von Visual Studio kann man sehr schnell in HTML-Dokumenten navigieren.



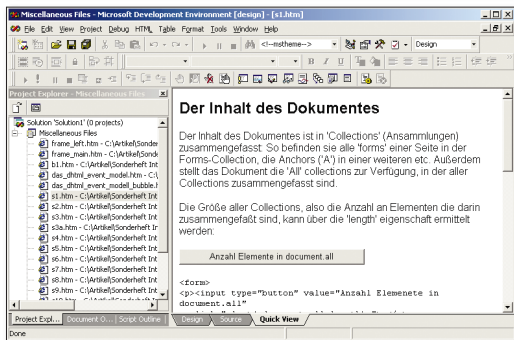
SPEZIALISIERTE TOOLS WIE VISUALINTERDEV sind für die Programmierung mit DHTML praktisch, aber keine Notwendigkeit.



Zum anderen ist es dadurch möglich den Inhalt einer Seite, unabhängig von der für die Seite durchzuführende Programmierung zu bearbeiten.

Ein externes Script wird über das SRC Attribut referenziert:

```
<SCRIPT LANGUAGE="JavaScript"
SRC="ExterneData1.js">
/*
hier könnte noch code eingefügt
werden der von browsern die das
SRC attribut nicht unterstützen
ausgeführt wird
*/
</SCRIPT>
```



BEI GRÖßEREN PROJEKTEN bieten die Projekt-Übersicht und die QuickView-Ansicht einen guten Überblick.

Das DHTML-Event-Modell

DHTML hat nicht nur ein Objektmodell, sondern auch ein Event-Modell. Das Event-Modell legt fest, welche Objekte innerhalb der Seite welche Ereignisse bearbeiten können.

Ein Ereignis (Event) tritt immer dann ein, wenn der Betrachter eines Dokuments eine Aktion durchführt oder wenn ein Vorgang ohne Einfluss des Anwenders ausgeführt wird. So gilt bei-

spielsweise jede Mausbewegung als Ereignis.

Genauso ist auch der Abschluss eines Download-Vorgangs ein Ereignis. Eine DHTML-Seite kann Programm-Code enthalten, der auf jedes dieser Ereignisse reagieren kann.

Auch ohne DHTML war es möglich einige Ereignisse mit Scripts zu behandeln. Allerdings ging das nur bei solchen Ereignissen, die die Entwickler des Browsers für „interessant genug“ gehalten hatten, um sie dem Programmierer der Webseite mitzuteilen. Bei DHTML kann jedes Ereignis behandelt werden, der Programmierer der Webseite kann für sich entscheiden, welches Ereignis interessant genug ist, um extra behandelt zu werden.

Für die Ereignisbehandlung sind zwei grundlegende Dinge relevant: das „Event Bubbling“ und die „Default Actions“.

Event Bubbling

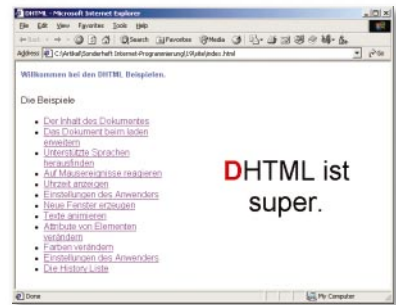
Unter „Event Bubbling“

versteht man die Art und Weise in der DHTML die Struktur des Dokuments verwendet, um auftretende Ereignisse an die einzelnen Objekte auf einer Webseite zu verteilen.

Ein DHTML-Dokument hat eine hierarchische Struktur, die sich aus einzelnen Objekten zusammensetzt. Bei DHTML lösen alle Objekte (Anchors, Paragraphen, etc.) Ereignisse aus bzw. erhalten „Event Notifications“ (Mitteilun-

gen über ein Ereignis), sofern es sie betrifft. Diese Mitteilungen werden zunächst an das direkt betroffene Element gesendet und dann an dessen Container-Element (Eltern-Element) bis hin zum Dokument.

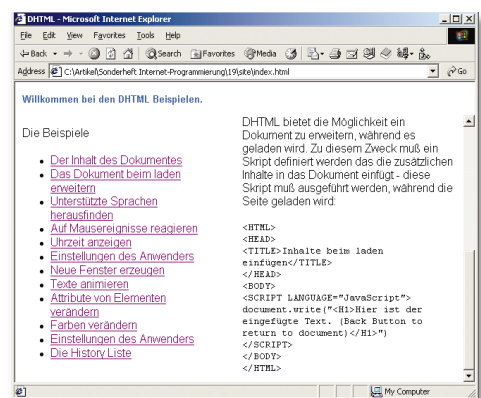
Dies bietet die Möglichkeit, ein Ereignis auf jeder Hierarchiestufe innerhalb des Dokuments zu behandeln; außer-



MIT DHTML KONNEN SIE zum Beispiel die Anzahl der Elemente auf einer Seite herausfinden. Das ist praktisch, wenn Sie darüber iterieren möchten.

dem ist es dadurch möglich, dass übergeordnete Elemente auf Ereignisse reagieren, die durch untergeordnete Elemente ausgelöst wurden.

Events werden durch alles mögliche ausgelöst: eine Mausbewegung, ein Mausklick, das Bewegen der Scrollbar und weiteres. Das Event-Modell sowie die Art und Weise, in der die Mitteilungen verschickt werden, entspricht weitgehend der Vorgehensweise, in der auch klassische Frameworks Messages weiterleiten.

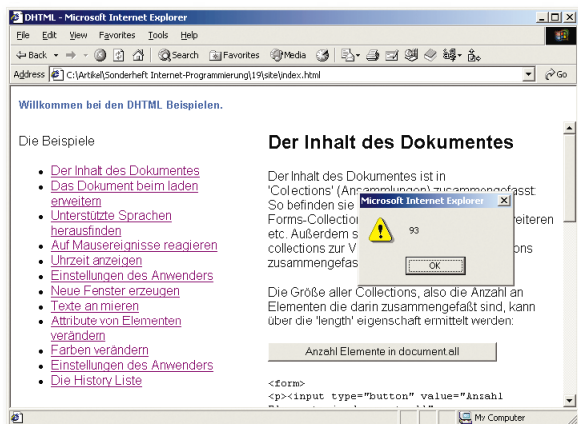


BEIM BEISPIEL-PROJEKT sind alle Quell-Codes am Ende jeder Seite im Quelltext aufgelistet.

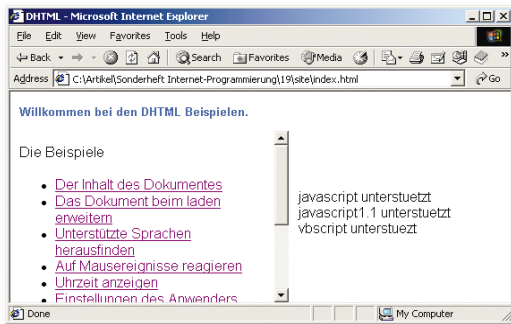
Default-Aktionen

Eine Default-Aktion ist die Aktion, die der Browser von sich aus ausführt, wenn ein bestimmtes Ereignis eintritt. Beispielsweise ist die normale vom Browser ausgeführte Aktion beim Klick auf eine HREF die, dass er dem Link folgt und die unter HREF angegebene Seite lädt und anzeigt.

Dabei ist die Default-Aktion nicht immer durch das auslösende Objekt definiert. Sie kann auch durch ein in der Hierarchie höher liegendes Element vorgegeben sein. Innerhalb der Event-Hierarchie kann jedes Objekt die De-



DAS BEISPIEL-PROJEKT ist eine komplette Website mit DHTML-Beispielen. Sie starten das Beispiel-Projekt über die Datei index.html.



DAS DOM KANN MIT verschiedenen Sprachen verwendet werden. Welche Sprachen vom Browser unterstützt werden, müssen Sie zunächst herausfinden.

fault-Aktion abbrechen. Sobald dies geschieht, wird die Aktion nicht durchgeführt.

Die Art und Weise mit der ein Ereignis mit einem Script assoziiert wird, nennt man Event Binding. DHTML bietet von sich aus verschiedene Arten des Event Bindings an. Außerdem kann eine Script-Sprache auch noch „selbst-gemachte“ Arten des Event Bindings anbieten. Beispiele für Event-Binding-Methoden sind „Event Attributes“, die ähnlich wie STYLE-Attribute vergeben werden, der „Generic“-Ansatz mit dem FOR-Attribut (siehe Beispiel) sowie der von VBScript verwendete Event-Binding-Mechanismus

Auf Mausereignisse reagieren

Mausbewegungen über dem Dokument lösen zum Beispiel ein MouseMove-Ereignis aus. Beim folgenden Beispiel wird das Script mit dem FOR-Attribut an das Dokument gebunden. Im Script wird die aktuelle Mausposition ermittelt und in der Statuszeile angezeigt.

```
<SCRIPT FOR="document"
  EVENT="onmousemove"
  LANGUAGE="JavaScript">
window.status = "X=" +
window.event.x + " Y=" +
window.event.y;
</SCRIPT>
```

Die Mitteilung über ein Ereignis ohne zusätzliche Informationen ist allerdings nur selten hilfreich. Zum Beispiel ist die Information, dass ein Mausklick ausgeführt wurde, nicht sinnvoll, wenn nicht bekannt ist, wo dieser Klick stattfand. Für solche Zusatzinformationen existiert das *event*-Objekt. Dieses enthält, wie im Beispiel-Code sichtbar, diese Informationen, etwa die Mauskoordinaten zum Zeitpunkt des Ereignisses.

Genau wie eine Mausbewegung löst auch ein Mausklick ein Ereignis aus. Im folgenden Beispiel wird der Titel der aktuellen Seite als Message-Box angezeigt, wenn der linke Maus-Button gedrückt wird:

```
<SCRIPT FOR="document"
  EVENT="onmousedown"
  LANGUAGE="JavaScript">
if(window.event.
  button==1)
alert(document.title);
</SCRIPT>
```

Das Window-Objekt

Das Browser-Fenster, innerhalb der Objekthierarchie als Window-Objekt zugänglich, ist das in der DHTML-Objekthierarchie am höchsten stehende Objekt. Mit Hilfe dieses Objekts können Sie Informationen über den Browser und die von ihm angezeigten Dokumente ermitteln.

Das Window-Objekt stellt unter anderem folgende Informationen und Objekte zur Verfügung:

- momentan angezeigte Frames,
- die URL des momentan angezeigten Dokuments,
- die Art und Version des Browsers sowie von diesem unterstützte Features (mit Hilfe des *navigator*-Objekts),
- Detailinformationen zu Ereignissen



MIT DHTML KÖNNEN SIE auch Dialogbox-ähnliche Fenster erzeugen.

(mit Hilfe des *event*-Objektes),

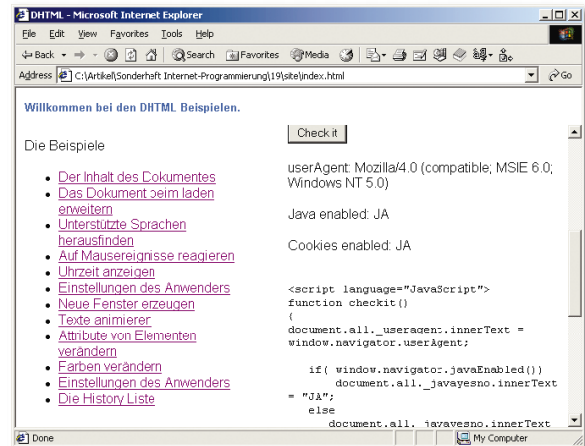
- das HTML-Dokument selbst (mit Hilfe des *document*-Objekts).

Mit dem Window-Objekt ist es möglich, die Fenster des Browsers selbst zu beeinflussen. Zum Beispiel kann der Inhalt des Statusfensters gesetzt werden:

```
<script language="JavaScript">
function StatusAendern()
{
    window.status="Bitte nicht
    nochmal klicken";
}
</script>

<form>
<input type="button" value="Click
me" onclick="StatusAendern();">
</form>
```

Das Window-Objekt bietet die Möglichkeit des Navigierens. Aus Sicherheitsgründen ist es zwar nicht möglich



BEIM PROGRAMMIEREN KÖNNEN SIE unter Umständen von den Einstellungen des Anwenders abhängig sein. Ob Cookies oder Java eingeschaltet sind, können Sie aber herausfinden und ggf. eine Warnung anzeigen.

die Liste der vom Anwender bereits besuchten URLs auszulesen, allerdings bietet DHTML dafür eine Möglichkeit, einen Mausklick auf den *Forward*- und *Back*-Button zu simulieren.

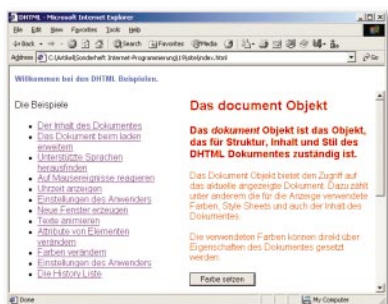
Dies geschieht mit Hilfe des *History*-Objekts. Dieses Objekt hat unter anderem die Methoden *back()* und *forward()*, die genau diesen Sinn erfüllen. Das folgende Beispiel implementiert zwei einfache Buttons, die die gleiche Funktionalität wie die Browser-Buttons haben:

```
<form name="nav_bar">

<input type="button" value="Zurück"
  onclick="window.history.back();">

<input type="button" value="Hin"
  onclick="window.history.
  forward();">

</form>
```

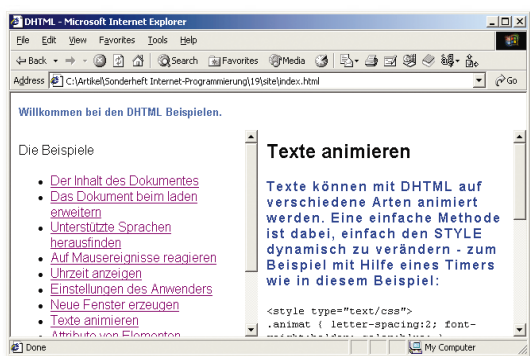



DIE FARBE DER ELEMENTE ist eines der Attribute, die Sie verändern können. Das geht auch interaktiv.

■ Timer – zeitabhängige Aktionen

Ein Timer-Event wird nach einem vorher festgelegten Zeitraum ausgelöst. Das Window-Objekt kennt zwei Arten von Timern: solche, die nur einmal nach einem bestimmten Zeitraum ausgeführt werden und solche, die immer wieder mit einem bestimmten zeitlichen Abstand ausgelöst werden.

Die erste Kategorie von Timern ist zum Beispiel dann hilfreich, wenn eine automatische Navigation implementiert werden soll. Eine Seite würde dann nur für einen bestimmten Zeitraum angezeigt, nach diesem Zeitraum wechselt der Browser automatisch zur nächsten Sei-



DURCH DIE VERÄNDERUNG von Größe und Farbe können Sie Text-Animationen gestalten. Das ist nicht immer hübsch, erregt aber Aufmerksamkeit.

te. Diese Funktionalität ist zum Beispiel für selbst ablaufende Demos interessant.

```
<script language="JavaScript">
var cSecs=25;
function AutoNavigate()
{
if( 0 == cSecs)
{
alert("Jetzt wurde die Seite
verlassen");
}
else
{
cSecs -= 1;
}
}
```

```
setTimeout("AutoNavigate()",
1000);
}
</script>
```

Die zweite Kategorie von Timern kann etwa für die Animation von Objekten verwendet werden. Im folgenden Beispiel wird ein solcher Timer verwendet, um innerhalb der Webseite eine Uhr darzustellen. Der Text, der die Uhrzeit darstellt, wird dazu einmal pro Sekunde erneuert.

Zu diesem Zweck ist es notwendig, dass das Script auf den tatsächlichen Text einer Seite zugreifen und diesen verändern kann. Dies geschieht mit Hilfe der `innerText`-Eigenschaft von Textelementen. Im Beispiel wird dazu ein Textkörper in Form eines SPAN-Elementes definiert und im Verlaufe des Scripts durch einen die aktuelle Uhrzeit beinhaltenden Text ersetzt.

Das eigentliche Script wird im Header der Seite eingebettet:

```
<script language=
"JavaScript">
function DisplayTime()
{
var time = new Date();
document.all.clock.
innerText =
time.getHours() + ":"
+ time.getMinutes() +
":" +
time.getSeconds();
}
</script>
```

In der Beschreibung des Körpers wird der Timer mit `setInterval()` gesetzt. Außerdem wird dafür gesorgt, dass der Timer beim Verlassen der Seite wieder zurückgesetzt wird:

```
<body onload=
"clearInterval
(window.tmr);"
onload="window.tmr =
setInterval
('DisplayTime()',
1000);">
```

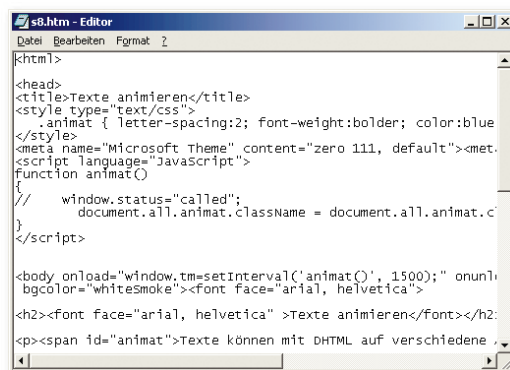
Im Körper des Dokuments wird zuletzt noch das SPAN-Element eingefügt, dessen Inhalt von dem oben stehenden Script verändert wird:

```
<p><span id="clock"><script
language="JavaScript">
document.write("Hier steht die
Zeit");
</script> </span></p>
```

Das Window-Objekt liefert auch Informationen über Einstellungen, die der Anwender innerhalb des Browsers vor-

genommen hat. Zu diesen Einstellungen gehört zum Beispiel, ob die Verwendung von Java und Cookies erlaubt sind. Unabhängig von diesen Einstellungen werden auch andere Informationen zur Verfügung gestellt, zum Beispiel Informationen über den Browser selbst:

```
<script language="JavaScript">
function checkit()
{
document.all._useragent.innerText
= window.navigator.userAgent;
if( window.navigator.
javaEnabled())
document.all._javayesno.innerText
= "JA";
else
document.all._javayesno.innerText
```



DEN QUELL-CODE aller Beispiele können Sie mit jedem beliebigen Editor ansehen, etwa mit Notepad.

```
= "Nein";
if(window.navigator.cookieEnabled)
document.all._cookiesyesno.
innerText = "JA";
else
document.all._cookiesyesno.
innerText = "Nein";
}
</script>
```

```
<form>
<p><input type="button"
value="Check it"
onclick="checkit();" > </p>
</form>
```

```
<p>userAgent: <span id=
_useragent"></span></p>
<p>Java enabled: <span id=
_javayesno"></span></p>
<p>Cookies enabled: <span
id="_cookiesyesno"></span></p>
```

■ Neue Fenster öffnen

Mit DHTML ist es auch möglich, neue Fenster zu erzeugen, und deren Position und Inhalt festzulegen. Eine typische „About“-Box kann beispielsweise dadurch definiert werden, dass ein neues modales Fenster geöffnet wird. Als Inhalt des Fensters verwendet man eine ganz normale Webseite:

```
<script language="JavaScript">
function AboutFxn()
{
window.showModalDialog
( "about.htm" );
}
```



```
}  
</script>  
  
<form>  
<p><input type="button" value=  
  "Info ueber..." onclick=  
  "AboutFxn();" > </p>  
</form>
```

■ Das Document-Objekt

Dieses Objekt ist für die Struktur, den Inhalt sowie den Stil des DHTML-Dokuments zuständig. Dazu zählen unter anderem die für die Anzeige verwendeten Farben, Style Sheets und auch der Inhalt des Dokuments.

Die verwendeten Farben können etwa direkt über Eigenschaften des Dokuments gesetzt werden. Im folgenden Beispiel wird bei jeder Betätigung des Buttons die Vordergrundfarbe (das ist die Farbe des Standardtextes) neu gesetzt:

```
<script language="JavaScript">  
var flag = 1;  
function ChgColor()  
{  
  if( 1 == flag) {  
    document.fgColor = "Red";  
    flag = 0;  
  }  
  else {  
    document.fgColor = "Blue";  
    flag = 1;  
  }  
}  
</script>  
  
<form>  
<p><input type="button"  
  value="Farbe setzen"  
  onclick="ChgColor();" > </p>  
</form>
```

■ Der Inhalt des Dokuments

Der Inhalt des Dokuments ist in „Collections“ (Ansammlungen) zusammengefasst. So befinden sich alle „forms“ einer Seite in der „Forms“-Collection, die Anchors („A“) in einer weiteren etc. Außerdem stellt das Dokument die „all“-Collections zur Verfügung, in der alle Collections zusammengefasst sind.

Die Größe aller Collections, also die Anzahl an Elementen, die darin zusammengefasst sind, kann über die *length* Eigenschaft ermittelt werden:

```
<form>  
<p><input type="button" value=  
  "Anzahl Elemente in  
  document.all"  
  onclick="alert  
  document.all.length);" > </p>  
</form>
```

Die einzelnen Elemente können auf verschiedene Arten angesprochen werden; hier sind in erster Linie die Features

der verwendeten Script-Sprache maßgebend. Im folgenden Beispiel wird die „item“-Methode der „all“-Collection verwendet, um die Bezeichner der einzelnen Collections anzuzeigen.

```
<script language="JavaScript">  
function ShowEl() {  
  for( var i=0;  
    i<document.all.length; i++)  
    alert( document.all.item(i)  
      .tagName);  
  }  
</script>
```

■ Verändern von Attributen

Die einzelnen Attribute von allen Elementen können verändert werden. Dies ist auf verschiedene Arten möglich, zum Beispiel durch das Setzen des „class“-Attributs. Das folgende Beispiel verändert etwa das class-Attribut des folgenden Elements, wenn sich der Maus-Cursor darüber befindet:


```
<style type="text/css">  
.yellow { background: yellow;  
font-weight: bolder}  
</style>  
  
<h1 onmouseover="this.className =  
'yellow';"  
onmouseout="this.className =  
'';">Dies ist der Teil des  
Dokumentes der sich  
verändert wenn sich die Maus  
darüber befindet.</h1>
```

■ Texte animieren

Texte können mit DHTML auf verschiedene Arten animiert werden. Am einfachsten ist es dabei, den STYLE eines Elementes dynamisch zu verändern, etwa mit Hilfe eines Timers wie in diesem Beispiel:

```
<style type="text/css">  
.animat { letter-spacing:2; font-  
weight:bolder; color:blue; }  
</style>  
  
<script language="JavaScript">  
function animat() {  
  document.all.animat.className =  
  document.all.animat.className ==  
  "animat" ? "" : "animat";  
}  
</script>  
  
<body onload="window.tm=  
  setInterval('animat()', 1500);"  
onunload="clearInterval(window.tm)  
>";>
```

DHTML bietet vielfältige Möglichkeiten zur Gestaltung interaktiver Webseiten, allerdings leidet es darunter, dass sich NetScape, Opera und Microsoft mal wieder nicht über die „richtige“ Implementierung einig sind. Es bleibt weiterhin zu hoffen, dass sich hier die Versprechungen der Hersteller irgendwann bewahrheiten und dass das

DOM wirklich ohne großen Aufwand in allen neueren Browsern eingesetzt werden kann.  UR

