Dynamic HTML

- **DHTML ist kein festgelegter Standard einer neuen Skript-Sprache, sondern einfach eine Kombination aus HTML, CSS, JavaScript und dem Document Object Model (DOM).**
- **Das Document Object Model ist der Versuch, alle relevanten Bestandteile einer WWW-Seite als Objekthierarchie abzubilden. Mit CSS kann das Aussehen und die Positionierung der Objekte kontrolliert werden und JavaScript erlaubt eine clientseitige Interaktivität.**
- **DHTML ist eigentlich nicht schwierig, nur die unterschiedliche Entwicklungen von Netscape und Microsoft erschweren das cross-browser-taugliche Angebote.**

CSS - weitere Möglichkeiten

- **Der Neuste Standard der Cascading Style Sheets (Version 2) bietet neben der Text-Formatierung auch Möglichkeiten zur Positionierung der Elemente im Browser-Fenster.**
- **Die 100%ige Unterstützung des Standards wird leider von keinem Browser geleistet.**
  - **Microsoft geht eigene Wege und Netscape 4.X ist nicht vollständig.**
- **Besonders interessant sind diese Möglichkeiten von CSS in Zusammenhang mit JavaScript, das diese Eigenschaften dynamisch verändern kann.**

CSS Positionierung

`position:`

- **Positionsart des beschriebenen HTML-Elements.**
- **Erlaubte Angaben:**
  - **absolute - absolute Positionierung, scrollbar**
  - **fixed - absolute Positionierung, scrollt nicht mit**
  - **relative - relativ, gemessen am Vorgängerelement**
  - **static - normaler Elementfluss, Standardeinstellung**
- **Außer der Positionierungsart ist auch die Angabe einer Startposition notwendig.**

JavaScript und HTML

- **eigenständige Programmiersprache**
- **Werkzeug zur Optimierung von WWW-Seiten**
- **Programmtext wird in der HTML-Datei oder in separaten Dateien notiert**
- **Quelltext wird zur Laufzeit interpretiert**
- **WWW-Browser besitzen entsprechende Interpreter-Software**
- **vergleichsweise einfache Sprache**
- **beschränkte Einsetzbarkeit (kein CGI-Script)**
- **unterschiedliche Behandlung in WWW-Browsern**

JavaScript und HTML

- **Mit der Verbindung von HTML, Cascading Style Sheets (CSS) und JavaScript wird es möglich dynamisches HTML (DHTML) zu entwickeln.**
- **DHTML erlaubt es während der Anzeige der Web-Seite einzelne Elemente dynamisch zu verändern.**
- **Die Entwicklung von browser- und plattform-übergreifenden DHTML-Lösungen erfordert die Sonderbehandlung der verschiedenen Browser.**

- **Jede nicht getestete Browser-Version funktioniert im Zweifel nicht!**

JavaScript = Sprache + Objektmodell

- **Korrektheit von JavaScript auf verschiedenen Browsern, verschiedenen Plattformen nicht nur abhängig von der Sprache.**
- **JavaScript als Sprache umfasst unter anderem Syntax, Datentypen und einige umgebungs-unabhängige Funktionen und Objekte. Hier hat sich von Browser zu Browser oder Version zu Version nicht viel geändert.**
- **Der zweite Teil des JavaScript-‘Systems‘ in einem Browser ist das Objektmodell (DOM), das für die Repräsentation des HTML-Dokuments im Browser existiert. Hier unterscheiden sich die verschiedene Browser wesentlich.**

Marc Schanne

7

Anwendungsgebiet von JavaScript

- **Steuerung des Browsers**
- **interaktive Web-Seiten**
- **Schnittstelle zu Java, um z.B. Netzzugriffe zu realisieren**

Marc Schanne

8

JavaScript in einer HTML-Datei

- **Üblicherweise JavaScript-Definition im Header der HTML-Datei.**
- **JavaScript kann aber an einer beliebigen Stelle der HTML-Datei definiert werden.**

```
<html><head>
    <script language="JavaScript">
      <!--
        alert("Hi!");
       //-->
    </script></head>
    <body></body>
</html>
```

Marc Schanne

9

JavaScript in einer separaten Datei

- **Im Header der HTML-Datei kann mit dem HTML-Tag** `<script>` **auch eine separate JavaScript-Datei mit JavaScript-Funktionen eingebunden werden:**

```
<html>
  <head>
    <script src="url"></script>
  </head>
 ...
```

Marc Schanne

1

JavaScript Datentypen

- **Zahlen**
- **Strings**
- **Boolesche Werte**
- **Funktionen**
- **Objekte**
- **Arrays**
- **Null-Wert**

JavaScript Funktionen

- **Funktionen haben Datentyp-Eigenschaften.**
- **Funktionen können in Variablen gespeichert werden.**
- **Vereinbarung von Funktionen über das Schlüsselwort** `function`**.**
- **Name einer Funktion ist ein Referenz auf die Funktion.**
- **Funktionen können mit mehr oder weniger Parametern aufgerufen werden, als im formalen Parametersatz angegeben.**
- **Funktionen sind im gesamten HTML-Dokument sichtbar.**

JavaScript
Objekte

- **werden durch `new`-Operator angelegt.**
- **`.` - Operator für Zugriff auf Member-Variablen.**
- **Objekte können andere Objekte beinhalten.**
- **Objekte können Methoden besitzen.**

JavaScript
Arrays

- **Array-Konstruktor `[]`**
- **Die Anzahl der in einem Array gespeicherten Elemente kann anhand der Eigenschaft `length` abgefragt werden.**

JavaScript
Definitionen von Variablen

- **Variablen sind in JavaScript nicht typsicher.**
- **Definition einer Variable mit dem Schlüsselwort `var` (nicht unbedingt nötig).**
- **Initialisierung**
  - **bei Definition**
  - **nachträglich.**

```
var monate;
monate=12;
tage=365;
var schaltjahr=true;
var stunden=24, minuten=sekunden=60;
```


Funktionen in JavaScript
Beispiel

- **Mit dem Schlüsselwort `function` wird die Definition einer Funktion eingeleitet.
  Dahinter folgt ein frei wählbarer Funktionsname und in Klammern `()` die Parameterliste.**
- **Der Inhalt der Funktion (Folge von Anweisungen) wird in geschweiften Klammern `{ ... }` angeschlossen.**

```
function name(Parameterliste){
   // JavaScript-Anweisungen
}
```

Funktionen in JavaScript

- **Funktionen sind abgeschlossene JavaScript-Prozeduren.**
- **Aufruf der Funktion über den Namen.**
- **JavaScript-Code, der nicht innerhalb einer Funktion steht, wird beim Einlesen der Datei vom WWW-Browser sofort ausgeführt.**

Funktionen in JavaScript

- **Funktion ist ein Anweisungsblock**
- **Funktionen können innerhalb**
  - **eines JavaScript-Bereichs oder**
  - **in einer separaten JavaScript-Datei definiert werden.**
- **Aufruf der Funktion:**
  - **innerhalb des einleitenden HTML-Tag** `<body ...>` **und bei Links** `<a href ...>`
  - **in einem Formular-Tag wie** `<input ...>` **(Ereignis)**
  - **innerhalb einer anderen Funktion**

Funktionen in JavaScript Beispiel

```
<html><head>
    <script language="JavaScript">
      <!--
         function trennstrich() {
           document.writeln(‘<BR><HR WIDTH="50%"‘ +
                            ‘ALIGN="center"></BR>‘);
         }
       // -->
     </script>
  </head><body>
    <h1>HTML-Seite</h1>
    <script>
      trennstrich();
    </script>
</body></html>
```

Marc Schanne

1

Ausführung einer JavaScript-Funktion

- **Ausführung wenn das Dokument geladen wird.**
- **Ausführung nach Bedarf:**
  - **aus einem Link**
  - **Ereignisgesteuert (z.B. beim Klick eines Buttons)**
- **Aufruf durch Angabe des Funktionsnamen mit den nötigen Parametern**
- **Eine Funktion kann einen ermittelten Wert an die aufrufende Instanz zurückgeben:** `return wert;`

Marc Schanne

2

Aufruf einer JavaScript-Funktion aus einem Link

```
<script language="JavaScript">
  <!--
      function simple() {
        var ergebnis = eval("1+1");
        alert("1 + 1 = " + ergebnis);
       }
    //-->
</script>
<body>
  Rechnen mit JavaScript-Funktion:
  <a href="javascript:simple()">Wieviel ist 1+1?</a>
</body>
```


Aufruf einer JavaScript-Funktion nach einem Ereignis

```
<html><head><title>Zahl auf Primzahl checken</TITLE>
  <script language="JavaScript">
    <!--
      function begruessen(zeichenkette){
        alert(zeichenkette);
      }
    //-->
  </script>
</head><body>
  <form name="Zeichekette">
    Meldung eingeben:<br> <input type="text" name="Meldung">
    <input type="button" value="Meldung..." onClick=
      "begruessen(document.PrimzahlFormular.Meldung.value)">
  </form>
</body></html>
```

JavaScript Objekte Grundlagen

- **JavaScript ist eine objektbasierte Sprache.**
  - **selbstdefinierte Objekte**
  - **im Browser vorhandene Objekte**
- **Objekte besitzen**
  - **Eigenschaften**
    - **Ansammlung von Daten gewisser Datentypen.**
  - **Methoden**
    - **Funktionen, die die Eigenschaften manipulieren oder mit ihnen umgehen.**

JavaScript Objekte - ein Beispiel

- **Schiff hat Eigenschaften**
  - **Name des Schiffes**
  - **Größe der Besatzung**
  - **maximale Ladekapazität**
  - **gegenwärtige Lademenge**
  - **Art der Ladung**

```
function schiff (name,besatzung,maxLadekapazität,aktLademenge,ladungsart){
  this.name = name;
  this.besatzung = besatzung;
  this.maxLadekap = maxLadekap;
  this.aktLademenge = aktLademenge;
  this.ladungsart = ladungsart;
}
```

JavaScript
Objekte - ein Beispiel

```
var cuttySark = new schiff("Cutty Sark", 39, 170, 53, 'tee');

document.writeln('Name des Schiffes: ' + cuttySark.name + '<BR>');
document.writeln('Besatzungsgröße: ' + cuttySark.besatzung + '<BR>');
document.writeln('Max. Ladekapazität: ' + cuttySark.maxLadekap + '<BR>');
document.writeln('Aktuelle Lademenge: ' + cuttySark.aktLademenge + '<BR>');
document.writeln('Art der Ladung: ' + cuttySark.ladungsart+ '<BR>');
```

- **Durch die Funktion** `schiff` **und durch die Verwendung des Schlüsselwortes** `new` **werden neue Schiff-Objekte erzeugt.**
- **Zugriff auf die einzelnen Eigenschaften des Objekts mit der . - Notation**

Besonderheiten von Objekten
Prototypen

Kreis.prototype

umfang = Kreis_umfang
pi = 3.14159

```
u = a.umfang();
f = a.pi * a.r *a.r;
b.pi = 4;
```

Kreis-Objekt: a

r = 1.0
x = 2.0
y = 3.0

Kreis-Objekt: b

r = 1.0
x = 2.0
y = 3.0
pi = 4;

Eingebaute Objekte

- **Vordefinierte Objekttypen**
  - `String`
  - `Array`
  - `Math`
  - `Date`
- **beinhalten eine Vielzahl von vordefinierten Methoden.**

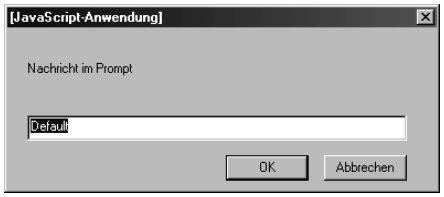
Client-Objekte

- **Client-Objekte beinhalten Informationen über die Umgebung in der das JavaScript abläuft.**
- **zwei Client-Objekte**
  - `navigator`**-Objekt**
  - `window`**-Objekt**

Interaktive Methoden

`alert()` **Nachricht vom Browser**

`confirm()` **liefert** `true` **oder** `false` **zurück (ok/cancel)**



`prompt()` **liefert die Eingabe des Anwenders zurück. Wenn nichts eingegeben wird "**`undefined`**"**

Marc Schanne

2

HTML-Formulare

- **HTML-Formulare beinhalten GUI-Elemente wie**
  - **Textfeld, Passwortfeld, Auswahlfläche, Schalter ...**
- **HTML-Formulare ermöglichen die Erstellung von interaktiven HTML-Seiten, indem Benutzereingaben an den Webserver übermittelt und dort verarbeitet werden.**
- **Der FORM-Tag mit unterschiedlichen INPUT-Tags ermöglicht diese Kommunikation.**
- **Die Übertragung der Daten an den Webserver wird durch Drücken einer SUBMIT-Schaltfläche gestartet.**
- **Rücksetzen auf ursprüngliche Werte mit einer RESET-Schaltfläche.**

Marc Schanne

3

HTML-Formularelemente JavaScript-Anknüpfung

- **Für den Zugriff aus JavaScript kann jedes HTML-Element durch das Attribut NAME identifiziert werden.**
- **Die wichtigsten JavaScript-Anknüpfungen bei den GUI-Elementen für die Ereignisbehandlung sind:**
  - **onBlur**
    Benutzer verlässt das Feld
  - **onClick**
    Benutzer klickt in das Feld
  - **onFocus**
    Benutzer aktiviert das Feld

Dynamische Erzeugung von Browserfenstern

- **Jedes** `window`**-Objekt besitzt eine Methode** `open()`**.**
- **Durch Parameter kann das Aussehen des geöffneten Fensters gesteuert werden.**

```
open(URL, fensterName[, Eigenschaften])
```

JavaScript & Ereignisse

- **JavaScript kann auf Ereignisse reagieren**
  - **Mausbewegung**
  - **Klick mit der Maus**
  - **Laden eines Dokuments**
  - **Fehler**
- **Zu manchen JavaScript-Objekten gibt es Event-Handler.**
- **Benutzung ist optional**

Event-Handler definieren

- **Event-Handler werden als zusätzliches Attribut innerhalb eines HTML-Tags definiert.**

```
<tag Event-Handler-Typ="JavaScript-Anweisungen">
```

- **Es empfiehlt sich Event-Handler nur dort zu verwenden wo sie von allen benutzen Browsers unterstützt und gleich verstanden werden!**

Event-Handler und Objekt Modell

- **Für den korrekten Einsatz von Event-Handlern muss neben den Unterschieden im Objekt-Modell der verschiedenen Browser (IE, Netscape, W3C) auch ein sehr unterschiedliches Verhalten beachtet werden.**
- **Welches Verhalten zeigt das folgende Beispiel?**

```
<html><body onClick= “alert(‘BODY‘);“>
  <div onClick=“alert(‘DIV‘);“>
    <a href=“#“ onClick=“alert(‘A‘);“>Link</a>
</body></html>
```


Event-Handler und Objekt Modell

- **In älteren Browsern wird durch Klick auf den Link direkt am Zielobjekt ein entsprechendes Klick-Event erzeugt.**
- **Beim Netscape 4 hingegen sickert das Event von oben nach unten durch die Objekthierarchie und kann auf dem Weg zum Zielobjekt schon abgefangen werden.**
  Mit der Definition der Funktion `klickHandler()` **und**
  `window.document.captureEvents(Event.CLICK);`
  `window.document.onClick=klickHandler;`
  kann z.B. ein Klick-Event schon auf Dokument-Ebene abgefangen werden.

Event-Handler und Objekt Modell

- **Der Internet Explorer kennt die verschiedene Event-Typen aus dem Objekt** `window.Event` **nicht. Außerdem bewegen sich die Events beim IE genau in umgekehrte Richtung. Im vorherigen Beispiel wird der Event-Handler am Ziel-Objekt zuerst ausgeführt und danach die der Vater-Knoten in der Dokumentstruktur.**
- **Mit dem IE 4 wurde auch ein** `event`**-Objekt eingeführt. Falls ein Event eintritt ist dies eine Eigenschaft von** `window` **und existiert nur vom Eintritt des Ereignisses bis zum Verlassen oben in der Elementhierarchie. Da nie 2 Events gleichzeitig exisitieren können reicht ein einziges** `event`**-Objekt aus.**

Event-Handler und Objekt Modell

- **Bei Netscape 6 und ebenso beim W3C gibt es kein Objekt namens** `window.event`**. Das** `event`**-Objekt wird automatisch erzeugt und als Parameter an die Handler-Funktion übergeben. Nur die Event-Verarbeitungsrichtung entspricht der im IE-Modell.**
- **Das Ausführen der Standardaktion eines Events kann nach dem Durchsickern bzw. Aufsteigen erfolgen. Wenn eine JavaScript-Funktion zur Behandlung des Events definiert wird, kann diese mit einem** `return` **die Ausführung der Standardaktion verhindern. Dies ist für unterschiedliche Events mal** `true` **oder** `false`**.**

Fenster und Ereignisse

- **Event-Handler im BODY-Tag eines Dokuments oder im FRAMESET-Tag definieren.**
- **onError()-Handler nur über eine Zuweisung angeben.**


```
<BODY onLoad="begruesse()" onUnload="verabschiede()">
```

```
window.onError=errorPopup;
```

Marc Schanne

3

Ereignisse - Übung

- **Verwenden sie den Event-Handler** `onMouseOver` **aus dem HTML-Tag A und wechseln Sie eine Grafik, die den Link darstellt.**
- **Anmerkungen:**
  - **mit** `new Image();` **erzeugen Sie ein Bild-Objekt und mit der Zuweisung an das** `src`**-Attribut können Sie die notwendigen Bilder vorladen.**
  - **Für das Ändern des Bildes braucht Ihr IMG-Tag einen Namen.**
  - `onMouseOut` **ist der Anknüpfungspunkt für das Verlassen des Links.**

Marc Schanne

4

DHTML

- **Dynamic HTML (DHTML) im weiteren Sinne sind schon dynamische Veränderungen wie MouseOver-Grafiken. Im engeren Sinne wird aber unter diesem Modewort die Möglichkeit verstanden, Elemente per Style Sheets ansprechen und manipulieren zu können.**
- **Die unterschiedliche Objektmodelle (OM) der verschiedenen Browser erschweren den Entwurf von plattformunabhängigem DHTML.**
- **Grundsätzlich gilt die Regel, dass nur getestete Browser-Versionen sicher unterstützt werden.**

Ebenen und JavaScript

- **Um von JavaScript aus auf mit CSS definierte Ebenen zugreifen zu können muss man die verschiedenen Objektmodelle der unterschiedlichen Browser beachten.**
- **Mit dem Navigator 4 wurde von Netscape** `layers[]` **als eigenständiges Objekte-Feld im** `document` **eingeführt. Jede Ebene stellt selbst wieder ein Dokument dar, hat deshalb eine eigene Eigenschaft** `document`**.**
  Beispiel-Zugriff auf `top`**-CSS-Attribut in Ebene** `dieEbene`**:**
  `self.document.layers['dieEbene'].document.top`
- **Für das Objektmodell des IE gibt es eine andere Zugriffsform. Zugriff über die** `all`**-Collection, über die auf alle Elemente gleichberechtigt zugegriffen werden kann.**

JavaScript Objektmodelle

MS Internet Explorer ab 4:

```
self.document.all['dasBild'].
src = "neueURL.gif";

self.document.all['dieEbene'].
style.top = '100px';
```

Netscape Navigator 4:

```
self.document.layers['dieEbene'].
document.images['dasBild'].src =
"neueURL.gif";

self.document.layers['dieEbene'].
top = 100;
```

Das W3C hat ebenfalls einen Standard für ein Objektmodell entwickelt, das 'Document Object Model' (DOM).

```
self.document.images['dasBild'].src = "imgN.gif";
self.document.getElementById("dieEbene").style.top="100px";
```


Browser-unabhängiges DHTML

- **In der Praxis gibt es drei verschiedene Objekt-modelle, die bei der Entwicklung von plattform-unabhängigem DHTML beachtet werden müssen.**
- **Überblick:**
  - **Navigator 4 → Netscape OM**
  - **IExplorer 4 → MS OM**
  - **IExplorer 5 & Netscape 6 → weitestgehend W3C DOM (Mozilla, Opera)**
- `if (document.layers) ... else if (document.all) ...` **nicht mehr ausreichend → „Crossbrowser-Programmierung“**

Crossbrowser-DHTML

- **100%ig sichere Browserabfragen gibt es leider nicht, 3 Möglichkeiten haben sich etabliert:**
  - `navigator`**-Objekt**
    Eigentlich sollte sich der Browser über Attribute `app???` **&** `userAgent` **eindeutig und richtig identifizieren.**
  - **Prüfung auf bestimmte Objekte** `document.layers` **existiert nur bei N4, aber evtl. können Browser zwar das Objekt kennen aber OM unvollständig implementiert. Bsp. iCab (Mac) kennt** `document.all` **aber nicht ähnlich zu IE.**
  - **Zugriff auf Objekt/Methode prüfen.**
- **In der Praxis ist eine Mischung aus erstem und zweiten Verfahren zu empfehlen.**

Crossbrowser-DHTML

```
userAgent = navigator.userAgent.toLowerCase();
n4 = document.layer;
ie = (document.all && userAgent.indexOf(“mac”) < 0);
w3c = document.documentElement;
dhtml = ((n4 || ie || w3c) && userAgent.indexOf(“aol”) < 0);
```

- **Browser, die man nicht wirklich mit der DHTML-Seite testen konnte sind grundsätzlich als unsicher einzustufen und besser gleich auf eine Ersatzseite ohne DHTML umzulenken. Umleitung mit JavaScript (**`location.replace()`**) oder mit dem META-Tag im Header.**
- **Natürlich kann auch jeder Browser auf eine eigene Zugangsseite umgeleitet werden.**

Crossbrowser-DHTML

- **Die DHTML-Fähigkeiten des N4 sind in vielen Bereichen die schwächsten → guter Ausgangspunkt für Entwurf.**
- **Nur Layer, mit CSS positionierte DIV- & SPAN-Elemente, können im N4 dynamisch verändert werden.**
- **Aufweitung für IE, dort sind alle Elemente dynamisch veränderbar.**
- **Anpassung an W3C von IE aus gering (kein `all[]`).**
- **Für kleine Skripts ist es durchaus akzeptabel bei jeder Anweisung einzeln zwischen den 3 Modellen zu verzweigen. Beispiel folgt …**

Crossbrowser-DHTML

Beispiel:

```
<div id="logo"
style="position:absolute;
left:10px; top:
10px">Logo</div>
```

```
if (document.layers)
  document.layers["logo"].
    visibility = "hidden";
else if (document.all)
  document.all["logo"].style.
    Visibility = "hidden";
else if
 (document.documentElement)
  document.
    getElementById("logo").
    style.visibilty = "hidden";
```

- **Entwurf für größere Projekt mit Crossbrowser-Funktionen:**
  Parametrisieren der Verzweigung mit einer `id` statt festem Element "`logo`". (Prozeduraler Entwurf)

Crossbrowser-DHTML
Beispiel: Uhrzeit

```
<html>
<head>
<title>Uhrzeit automatisiert</title>
</head>
<body>

Die Zeit, es ist:
<div id="uhr" style="position:absolute; left:10; top:30">
</div>

<script language="JavaScript1.2">
<!--
  function tick() {
   var jetzt = new Date();
   var stunden = jetzt.getHours();
   var minuten = jetzt.getMinutes();
   var sekunden = jetzt.getSeconds();
   if (stunden < 10) stunden = "0"+stunden;
   if (minuten < 10) minuten = "0"+minuten;
   if (sekunden <10) sekunden = "0"+sekunden;
   var zeit = stunden+":"+minuten+":"+sekunden;
```

Marc Schanne

4

Crossbrowser-DHTML
Beispiel: Uhrzeit

```
   if (document.layers) {
      alert("Netscape 4.X");
      document.layers["uhr"].document.write(zeit);
      document.layers["uhr"].document.close(); }
   else if (document.all) {
      alert("IE");
      document.all["uhr"].innerHTML = zeit; }
   else if (document.getElementById) {
      alert("W3C");
      document.getElementById("uhr").firstChild.nodeValue = zeit;
   }
}
setInterval("tick()",1000);
// -->
</script></body></html>
```

Marc Schanne

5