Aktuelle Webservertechniken

JavaScript / ECMAScript

Referenzen:
– Netscapes Java Script online Reference Manual
http://developer.netscape.com/docs/manuals/js/client/jsguide/index.htm
http://developer.netscape.com/docs/manuals/js/client/jsref/index.htm
– Voodoos Intro to JavaScript
http://rummelplatz.uni-mannheim.de/~skoch/js/tutorial.htm
– Stefan Münz / selfhtml

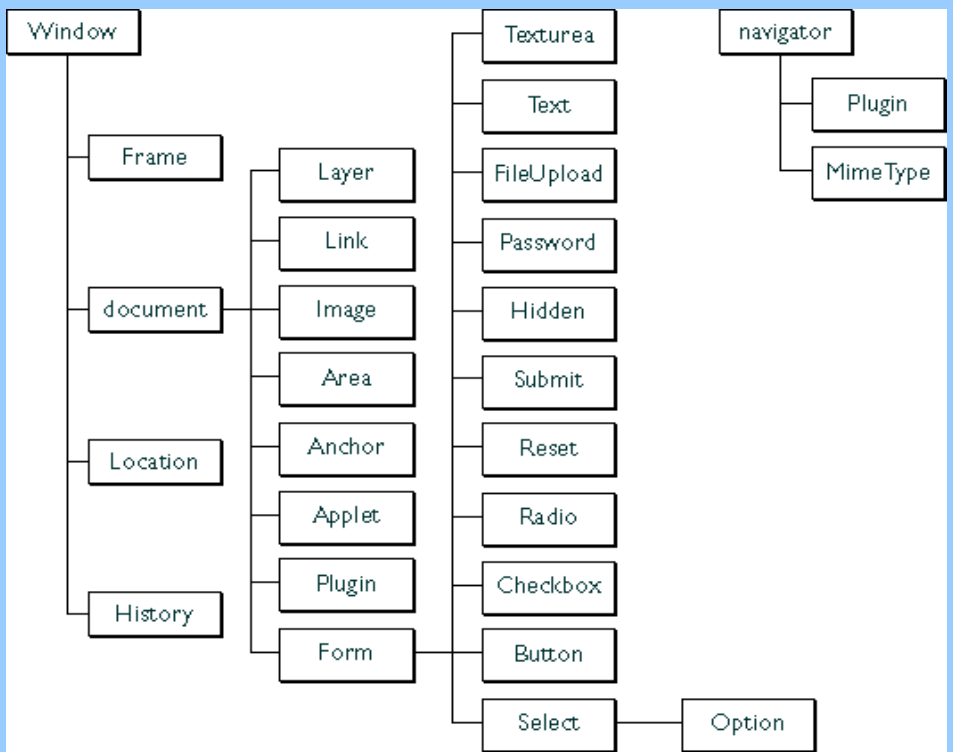
Zuordnung, Abgrenzung

- ➔ Einfache Skriptsprache, eingeführt von Netscape mit Version 2.0, ca. 1996 (?), von anderen Browsern schnell übernommen. Heute implementiert JavaScript wie JScript ECMAScript (*European Computer Manufatures Association*)
- ➔ Kein Java – aber im Zuge der Java–Hype so benannt.
- ➔ Ähnlicher Zweck wie Java–Applets: Programme, die im Browser des Anwenders laufen. Dynamisierung der Webseite lokal.

Einordnung

- ➔ Interpreted (not compiled) by client.
- ➔ Object–oriented. No distinction between types of objects. Inheritance is through the prototype mechanism, and properties and methods can be added to any object dynamically.
- ➔ Code integrated with, and embedded in, HTML.
- ➔ Variable data types not declared (dynamic typing).
- ➔ Cannot automatically write to hard disk.

Dokumentenstruktur aus JS–Sicht



JavaScript in HTML

```
<html>
<body>
<br>
This is a normal HTML document.
<br>
  <script language="JavaScript">
  <!-- hide from old browsers

    document.write("This is JavaScript!")

  // -->
  </script>
<br>
Back in HTML again.
</body>
</html>
```

Achtung: im Unterschied zu php ist das das, was der Browser sieht, vom Server wird es nicht interpretiert!

JavaScript in HTML

```
<html>
<head>

<script language="JavaScript">
<!-- hide

function calculation() {
  var x= 12;
  var y= 5;
  var result= x + y;

  alert(result);
}

// -->
</script>

</head>
<body>

<form>
<input type="button" value="Calculate" onClick="calculation()">
<input type="button" value="Click Mich"
onClick="location.href='http://wwwmath.uni-muenster.de/u/lammers';">

</form>
</body>
```

Events (Auszug)

Event(Handler)	Bezug	Auftreten
OnAbort	Bilder	Benutzer bricht das Laden eines Bildes ab.
OnFocus	Fenster, forms	Element bkommt den Eingabefokus
OnKeyPress	Fast überall	Benutzer drückt einen Key
OnKeyDown	Fast überall	Entsprechend
OnKeyUp	Fast überall	Entsprechend
OnLoad	Body	Beim Laden eines Dokuments
OnMouseDown	Doc, buttons, links	Mausknopf wird gedrückt
OnMouseOver	Doc, buttons, links	Mauszeiger über dem Objekt
OnSelect	Textfelder	User hat das Feld selektiert
OnSubmit	Forms	Submit gewählt
OnUnload	Doc body	Dokument wird verlassen
OnClick	Buttons	Button wird angeclickt
OnReset	Forms	Reset button wird geclickt


```
<HEAD>
<SCRIPT>
<!--- Hide script from old browsers
function compute(f) {
   if (confirm("Are you sure?"))
      f.result.value = eval(f.expr.value)
   else
      alert("Please come back again.")
}
// end hiding from old browsers -->
</SCRIPT>
</HEAD>

<BODY>
<FORM>
Enter an expression:
<INPUT TYPE="text" NAME="expr" SIZE=15 >
<INPUT TYPE="button" VALUE="Calculate" onClick="compute(this.form)">
<BR>
Result:
<INPUT TYPE="text" NAME="result" SIZE=15 >
</FORM>
</BODY>
```

Zugriff auf Browser und Dokument

Sehr viele Elemente stehen als Arrays zur Verfügung (vgl. Doc–Struktur), u.a.

```
document.anchors, -.applets, -.forms, -.images, -.layers, -
   .links

Form.elements

navigator.plugins

window.history
```



```
...
<center>
  [<a href="Script/">Script</a>]
  [<a href="Projekt1/">CGI-BSP</a>]
</center>

<script>
document.write("<br>Linkliste:");
for (var i = 0; i < document.links.length; i++) {
 document.write("<br>" + i + ": ");
 document.write(document.links[i]);
}
</script>

<hr>
<address>
...
```

Sprachbasis

- Syntax vergleichbar Java entsprechend auch die Kontrollstrukturen (*if–then–else*, *switch*, *while,* ...) und die Objektreferenzierungen.
- Basistypen (Konstanten): Zahlen, Strings, *true*, *false*, *null* und *undefined*
- Variablen können beim ersten Auftreten mit *var* deklariert werden. Ansonsten muss das erste Auftreten eine Wertzuweisung sein. In Funktionsdeklarationen ist *var* obligatorisch, Variablen dort sind lokal. Globale Variablen können auch über das window–Objekt von „aussen“ referenziert werden.

Funktionen

- Funktionen werden wie üblich definiert und benutzt – natürlich in der ungetypten Variante.

```
function square(number) {
    return number * number;
}
```

- Argumente werden implizit im Array *fname.arguments* übergeben, man kann damit variable Argumentlisten aufbauen

```
function myConcat(separator) {
   result="" // initialize list
   // iterate through arguments
   for (var i=1; i<arguments.length; i++) {
      result += arguments[i] + separator
   }
   return result
}
```

Arrays und Objekte

- Objekte sind assoziative Arrays:

```
objekt.wert  ==  objekt["wert"]
```

- Objekte kann man mit Konstruktorfunktionen erzeugen:

```
function person(name, sex) {
   this.name = name;
   this.sex = sex;
}
ich = new person("lammers","m");
du = new person("wuebbel","m");
```


Input / Output

- ➔ Input über formfeld–values oder *window.prompt*
- ➔ Output über diverse Methoden, u.a.
  - ➔ *alert("Nachricht")* und *confirm("Nachricht")*
  - ➔ *document.write("Text")*
  - ➔ In der Statuszeile mit *window.status="text"*
  - ➔ ...
- ➔ Kein Datei–I/O

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.2//EN">
<html>
  <head>
    <script>
      <!--- Hide script from old browsers
      var text = "Ha";
      var iid = 0;

      function mehr () {
        text += "a";
        document.Demo.expr.value=text;
      }
      function reicht () {
        text += "tschie ...";
        document.Demo.expr.value=text;
        window.clearInterval(iid);
      }
      // end hiding from old browsers -->
    </script>
  </head>

  <body>
    <h2>Gesundheit!</h2>
    <script>
      confirm(prompt("Oder?","Etwa nicht!!"));
    </script>

    <form name="Demo">
      <input type="text" name="expr" size="65"
             onMouseOver="window.status='Das ist der Renner'; return true;">
      <script>
        mehr();
        iid = window.setInterval(mehr,1000);
      </script>
      <input type="button" value="schluss" onClick="reicht();">
    </form>
    <br>Last modified: <script>document.write(document.lastModified)</script>
  </body>
</html>
```



```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.2//EN">
<html>
  <head>
    <script>
      <!--- Hide script from old browsers
      var text = "Ha";
      var iid = 0;

      function mehr () {
        text += "a";
        document.Demo.expr.value=text;
      }
      function reicht () {
        text += "tschie ...";
        document.Demo.expr.value=text;
        window.clearInterval(iid);
      }
      // end hiding from old browsers -->
    </script>
  </head>
```

```
<body>
   <h2>Gesundheit!</h2>
   <script>
     confirm(prompt("Oder?","Etwa nicht!!"));
   </script>

   <form name="Demo">
     <input type="text" name="expr" size="65"
            onMouseOver="window.status='Das ist der Renner';
            return true;">
     <script>
     mehr();
     iid = window.setInterval(mehr,1000);
     </script>
     <input type="button" value="schluss" onClick="reicht();">
   </form>
   <br>Last modified:
        <script>document.write(document.lastModified)</script>
  </body>
</html>
```


Einsatz

- Vorvalidierung von Eingaben in *forms,* ggf. Datensammlung und Vorauswertung (z.Bsp. Endsumme im Warenkorb, Calculator)
- Elegantere Benutzerführung und –hilfe, autom. Einfügen aktueller Daten (Datum, Uhrzeit etc.)
- Dynamische Webseiten, insbesondere bei Einsatz von *Layern* (abs. + rel. Offset, unsichtbar oder sichtbar, etc.)
  http://rummelplatz.uni–mannheim.de/~skoch/js/part10/part10.htm#transp , http://www.netzwelt.com/selfhtml/tedga.htm
- Risiko: auf Benutzer/Browserseite. Unleugbar da. Es gibt Security–Konzepte: *same origin, signed scripts*
- Auch als ServerSide–Scriptsprache einsetzbar (nur netscape–Server?)