10. Internet-basierte Digitale Medien

Weiterführende Literatur:

Wolfgang Dehnhardt: JavaScript, VBScript, ASP, Perl, PHP, XML: Scriptsprachen für dynamische Webauftritte, Carl Hanser 2001

http://de.selfhtml.org/

Gliederung

1. Grundbegriffe
2. Digitale Codierung und Übertragung
3. Zeichen und Schrift
4. Signalverarbeitung
5. Ton und Klang
6. Licht, Farbe und Bilder
7. Bewegtbilder
8. Animation und Interaktion
9. Mediendokumente
10. Internet-basierte digitale Medien
11. Computergrafik und Virtuelle Realität

Hinweis auf thematische Einschränkung

- Hier wird ausschliesslich die Frage der dynamischen Erzeugung von in Web-Browsern dargestellten Informationen diskutiert.
  - Nur ein Aspekt von Internet-basierten Digitalen Medien
  - Orientierung auf Kommunikationsmedium im Sinn von Foren, E-Commerce, E-Government etc.

- Weitere Aspekte:
  - “Streaming” von gespeicherten oder in Realzeit erzeugten Audio- oder Videodaten
  - Videokonferenzen
  - ...

Was ist JavaScript?

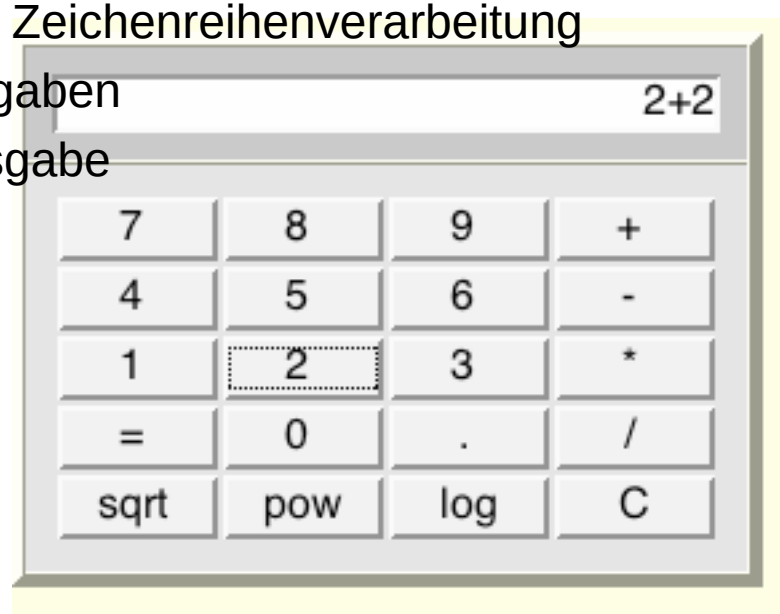
- Schlanke Programmiersprache zur integrierten Ausführung in Web-Browsern (und -Servern)
  - interpretiert
  - lokale Ausführung ohne weitere Kommunikation
  - objektbasiert (nicht echt objektorientiert, z.B. keine Klassen/Vererbung)
  - schwach typisiert
  - dynamisch gebunden
  - relativ sicher (kein Zugriff auf lokales Dateisystem und Betriebssystem)
- JavaScript hat ausser einer gewissen Syntaxähnlichkeit keine Beziehung zu Java! (Originalname: "LiveScript")
- Geschichte:
  - Entwickelt von Netscape1995 (ab Browserversion 2)
  - Unterstützung in Microsoft Internet-Explorer ab Version 3 ("JScript")
  - Standardisiert als ECMAScript (ECMA-262) (European Computer Manufacturers Association) bzw. als ISO-10262
  - Moderne Browser weitgehend kompatibel zum ECMA-Standard

Skriptsprachen allgemein

- Enge Integration mit Betriebssystem oder speziellem Anwendungssystem
- Meist interpretiert, dadurch leicht zur Laufzeit zu definieren und zu ändern
- Moderne Skriptsprachen durchaus Alternative zu Programmiersprachen
- Beispiele:
  - Betriebssystem-Skripte: Unix Shells, DOS Batch-Dateien, AppleScript
  - Clientseitige Web-Skripte: JavaScript, VBScript
  - Serverseitige Web-Skripte: PHP
  - Skripte für Multimedia-Player: Flash ActionScript
  - Universelle Skripte: Perl, Python

JavaScript: Funktionsumfang und Anwendungsbereich

- Beispiele für sinnvolle Anwendung von JavaScript:
  - Formulareingaben auf Plausibilität prüfen
  - Spezialitäten verschiedener Browser-Plattformen flexibel unterstützen ("Browser-Weichen")
  - Bei Einbindung von Multimedia-Datei überprüfen, ob Browser ein Format unterstützt
- Funktionsumfang:
  - Klassische Funktionen für Arithmetik und Zeichenreihenverarbeitung
  - Verarbeitung von Maus- und Tastatureingaben
  - Dynamische Erzeugung von (HTML-)Ausgabe
  - Zugriff auf Dokument-Struktur über das *Document Object Model (DOM)*



Dynamisches HTML (DHTML)

- Kein wirklich genau festgelegter Begriff!

- Nach W3C korrekte Bedeutung:
  - HTML
  - Cascading Style Sheets (CSS2)
  - JavaScript/ECMAScript
  - DOM

- Verbreiteter Sprachgebrauch:
  - Jede Technik, bei der Web-Seiten ihren Inhalt abhängig von Benutzereingaben oder Zeitverlauf ändern

    (auch serverseitige Berechnung von HTML)

Einbettung von JavaScript in HTML

```
  <h1>
<!-- Script-Markup -->
      <script type="text/javascript">
            document.write("Hello World!");
      </script>
  </h1>
<!-- Externe Datei -->
  <h2>
      <script type="text/javascript" src="hello.js"></script>
  </h2>
<!-- URI -->
  <h2>
      <a href="javascript:alert('Hallo');">Hallo sagen</a>
  </h2>
<!-- Eventhandler -->
  <h2 onClick="confirm('Halli');">
      Hier klicken...
  </h2>
```

Browser ohne JavaScript-Unterstützung...

```
<script type="text/javascript">
     document.write("Hello World!");
</script>

<noscript>
     <i>Bitte m&ouml;glichst JavaScript
          einschalten, danke.</i>
</noscript>
```

- Der Inhalt des <noscript>-Tags wird ausgegeben, wenn JavaScript deaktiviert ist.
- Hinweis: Um auch Browser zu berücksichtigen, die das <script>-Tag nicht erkennen, wird das Skript oft zusätzlich in einen HTML-Kommentar eingeschlossen.

JavaScript: Kommentare, Namen, Literale

- Kommentarzeilen:
  - beginnen mit **`//`** oder werden in **`/* ...*/`** eingeschlossen
  - **`<!--`** *ist ein spezieller einzeiliger Kommentar.*
- Variablennamen beginnen mit Buchstaben, Dollar oder Unterstrich
- Gross- und Kleinschreibung wird unterschieden
- Numerische Literale (Beispiele):
  - Dezimale Ganzzahlen: **`0, 22, -1000`**
  - Oktalzahlen mit **`0`** beginnend: **`026`** (= dezimal 22)
  - Hexadezimalzahlen mit 0x beginnend: **`0x16`** (= dezimal 22)
  - Fließkommazahlen: **`33.333, 123., 6.24e-12`**
- Zeichenreihen-Literale:
  - Wahlweise in *einfachen oder doppelten* Anführungszeichen
  - Sonderzeichen **`\b, \n, \t, ...`**

- *Sehr ähnlich zu, aber nicht identisch mit Java-Syntax*

Skripte und Kommentare

- Für Browser, die die Skriptsprache JavaScript nicht erkennen:
  - JavaScript in HTML-Kommentar einschließen
  - HTML-Kommentarzeichen für JavaScript auskommentieren
    - » Spezieller JavaScript-Kommentarbeginn **<!--**
- Beispiel:

```
<script type="text/javascript">
  <!--
     document.write("Hello World!");
  // -->
</script>

<noscript>
     <i>Bitte m&ouml;glichst JavaScript
          einschalten, danke.</i>
</noscript>
```

Schwache Typisierung

- Jede Variable und jeder Funktionsparameter kann uneingeschränkt Werte eines jeden in JavaScript bekannten Datentyps annehmen:
  - Zahl (Ganzzahl, Fließkomma)
  - Zeichenreihe
  - Wahrheitswert
  - Array
  - Objekt
  - Funktion
- Ergebnisse von Funktionen werden mit **`return`** übergeben; ebenfalls keine Typdeklaration
- Variablendeklaration:
  - explizit: **`var i; var i = 1;`**
  - implizit bei Verwendung: **`i = 12;`**
- Abfrage des aktuell zugewiesenen Datentyps:
  - **`typeof v`**

Programm-Beispiel: Fibonacci-Funktion

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

function fib(n){
  if (n==0)
     return 0;
  else
     if (n==1)
          return 1;
     else
          return(fib(n-1)+fib(n-2));
}

document.writeln("fib(3) ="+fib(3)+"<br>");
document.writeln("fib(8) ="+fib(8)+"<br>");

</script>
```

Arrays (Felder) in JavaScript

- Indizierte Arrays:
  - Inhalt wie üblich über Zahl-Index adressiert

    `a = new Array(1, 2, 3, "vier");`

    `a = ["one", 2.1, , 4];`

    Lesen: **`a[0]  a[3]`**

- Assoziative Arrays:
  - Inhalt Schlüssel-Wert-Paare, über Schlüssel adressiert
  - Wahlweise in Array-Syntax oder Attribut-Syntax

    `a = new Array();`

    `a["x"] = "y";` Lesen: **`a["x"] a.x`**

    `a = {"x":"X", "y":"Y"};` Lesen: **`a["x"]  a.y`**

Programm-Beispiel zu Variablen und Feldern

```
function show(a){
  document.writeln("a: "+a); document.writeln("<br>");
  document.writeln("a[0]: "+a[0]); document.writeln("<br>");
  document.writeln("a[1]: "+a[1]); document.writeln("<br>");
  document.writeln("a[2]: "+a[2]); document.writeln("<br>");
  document.writeln("a[3]: "+a[3]); document.writeln("<br>");
  document.writeln("<hr>");
}

var a = new Array(1, 2, 3, 4); show(a);
a[2] = "drei"; a[3] = 4.01; show(a);

a = {"Strasse":"Amalienstr.", "Nr":17,
     "Ort":"M&uuml;nchen", "PLZ":80333};
document.writeln(a.Strasse+" "+a.Nr+"<br>");
document.writeln(a.PLZ+" "+a.Ort+"<br>");
```

Zeichenreihen (Strings)

- Viele vordefinierte Eigenschaften und Funktionen, z.B.:
  - **`length`**: Länge der Zeichenreihe
  - **`concat`**: Verkettung von Zeichenreihen
  - **`indexOf`**: Position einer Teilzeichenreihe
  - **`substring`**: Ausschneiden einer Teilzeichenreihe
  - **`search, match, replace`**: Suchen und Ersetzen von Teilzeichenreihen, die über *reguläre Ausdrücke* spezifiziert sind (z.B. **`/dm.*/`**)
- Aufruf in "objektorientiertem" Stil: *Objekt* **`.`** *Funktion*
- Detaillierteres Beispiel:
  - **`split(`*`begrenzer`*`)`**: Teilt Zeichenreihe in ein Array von Teilzeichenreihen gemäß dem Trennzeichen *begrenzer*

```
s = ("Fritz;Eva;Franz;Maria");
a = s.split(";");
```

ergibt
a =**`["Fritz","Eva","Franz","Maria"]`**

Programm-Beispiel zu Zeichenreihen und Schleife

```
var s = "Das ist ein Beispieltext f&uuml;r Strings in
JavaScript";
document.writeln(s+"<br>");
a = s.split(" ");
document.writeln(a+"<br>");


d = 0;
for(i = 0; i < a.length; i++) {
     d = d + a[i].length;
}
d = d / a.length;
document.writeln("Durchschnittliche Wortl&auml;nge:
"+d+"<br>");
```

Ablaufstrukturen in JavaScript

- Ablaufsteuerung ist analog zu Java-Syntax und -Semantik, z.B.:
  - if/else
  - for
  - while
  - switch
  - return
  - break
  - continue

Exkurs zu HTML: Formulare

- Benutzereingabe in HTML:
  <form>-Tag
- Untergeordnete Tags:
  - **<input type=*typ* name=*name*>**

    Mögliche Typen (*typ)* (Auswahl):

checkbox	Wahl-Kästchen
radio	"Radio-Knöpfe" für Alternativen
text	Textzeile
textarea	Mehrzeiliges Textfeld
password	Textfeld zur Passwortabfrage
file	Dateiauswahl
button	Allgemeine Schaltfläche
submit	Schaltfläche zum Absenden des Formularinhalts
reset	Schaltfläche zum Zurücksetzen des Formularinhalts

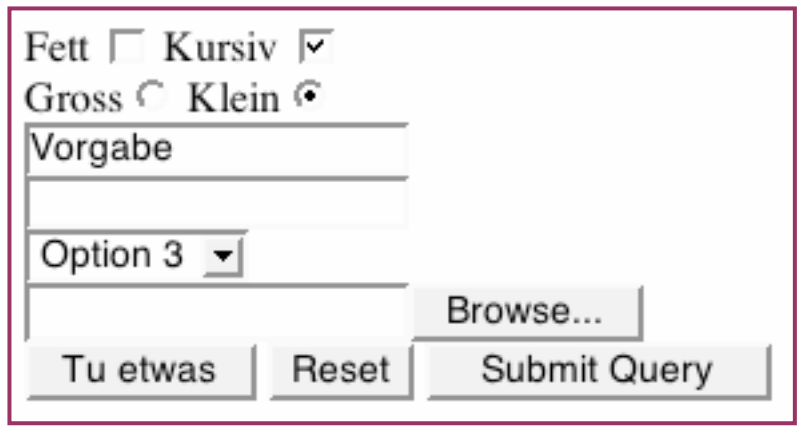
  - **<select name=*name*>**

    Liste von Optionen: **<option>**

    <option selected> bestimmt "vorselektierten" Standardwert

Beispiel: HTML-Formular

```
<form>
   Fett <input type="checkbox" name="cb" value="fett">
   Kursiv <input type="checkbox" name="cb" checked
       value="kursiv"><br>
   Gross<input type="radio" name="rad" value="gross">
   Klein<input type="radio" name="rad" value="klein" checked>
   <br>
   <input type="text" name="txt" value="Vorgabe"><br>
   <input type="password"><br>
   <select name=sel">
       <option>Option 1</option>
       <option>Option 2</option>
       <option selected>Option 3</option>
   </select><br>
   <input type="file" name="fil"><br>
   <input type="button" name="button1"
       value="Tu etwas">
   <input type="reset">
   <input type="submit">
</form>
```



JavaScript-Funktionen für modale Dialoge

- Dialogtypen:
  - *modal:* System geht in neuen Zustand und wartet auf Antwort, bevor normale Verarbeitung fortgesetzt wird
    - » Typisches Beispiel: Öffnen-Dialog mit Dateiauswahl
  - *nicht-modal:* Dialogbearbeitung wird parallel und unabhängig zur normalen Arbeit des Systems fortgeführt
    - » Typisches Beispiel: Objektinspektor in Entwicklungsumgebungen
- Standardtypen von modalen Dialogen:
  - Hinweis:
    - » System will sicherstellen, dass bestimmte Information vom Benutzer wahrgenommen wurde. Meistens ein "OK"-Knopf

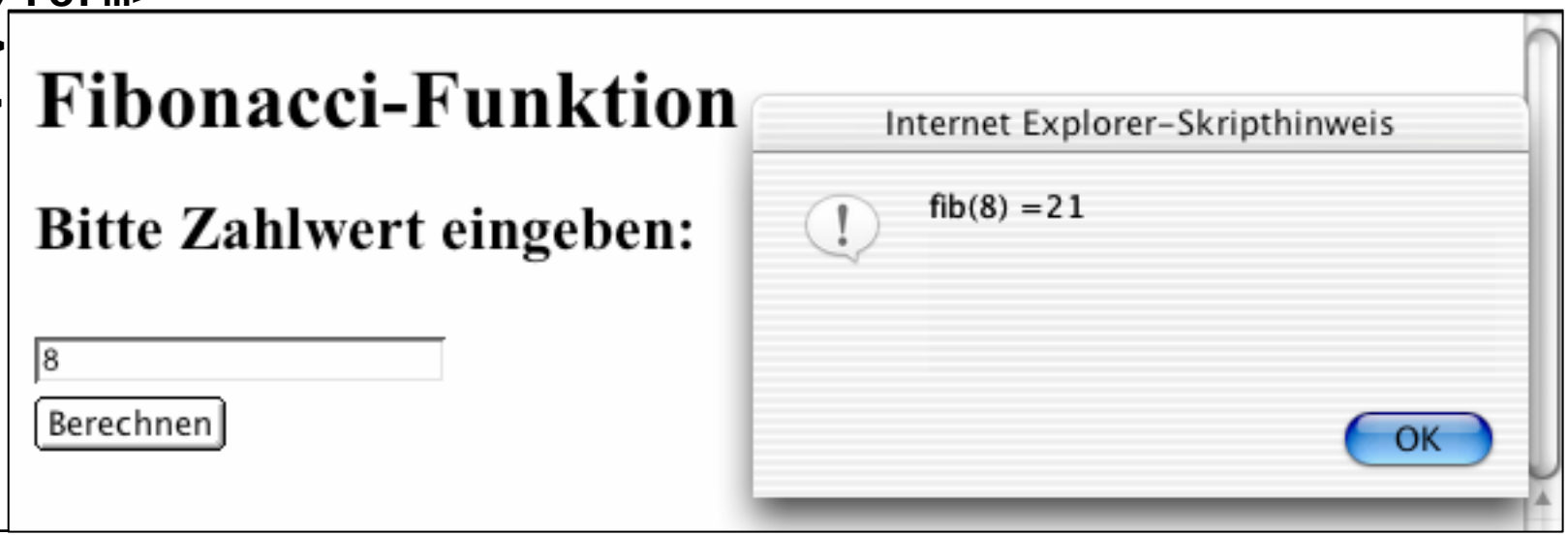
      JavaScript: **`alert(`*`String`*`)`**
  - Bestätigung:
    - » System will für eine bestimmte Entscheidung eine Bestätigung oder Ablehnung vom Benutzer erhalten. Meistens "OK"- und "Cancel"-Knopf

      JavaScript: **`confirm(`*`String`*`)`**
  - Abfrage:
    - » System will eine bestimmte Eingabe vom Benutzer erhalten.

      JavaScript: **`prompt(`*`String, StandardwertString`*`)`**

Beispiel: Fibonacci-Programm mit HTML-Eingabe

```
<body>...
  <h2>
      Bitte Zahlwert eingeben:
      <form name="formular">
            <input type="text" name="eingabe" value="0"><br>
            <input type="submit" value="Berechnen"
              onClick="
               var eing = document.formular.eingabe.value;
               alert('fib('+eing+') ='+fib(eing));">
      </form>
  </h2>
</body>
```



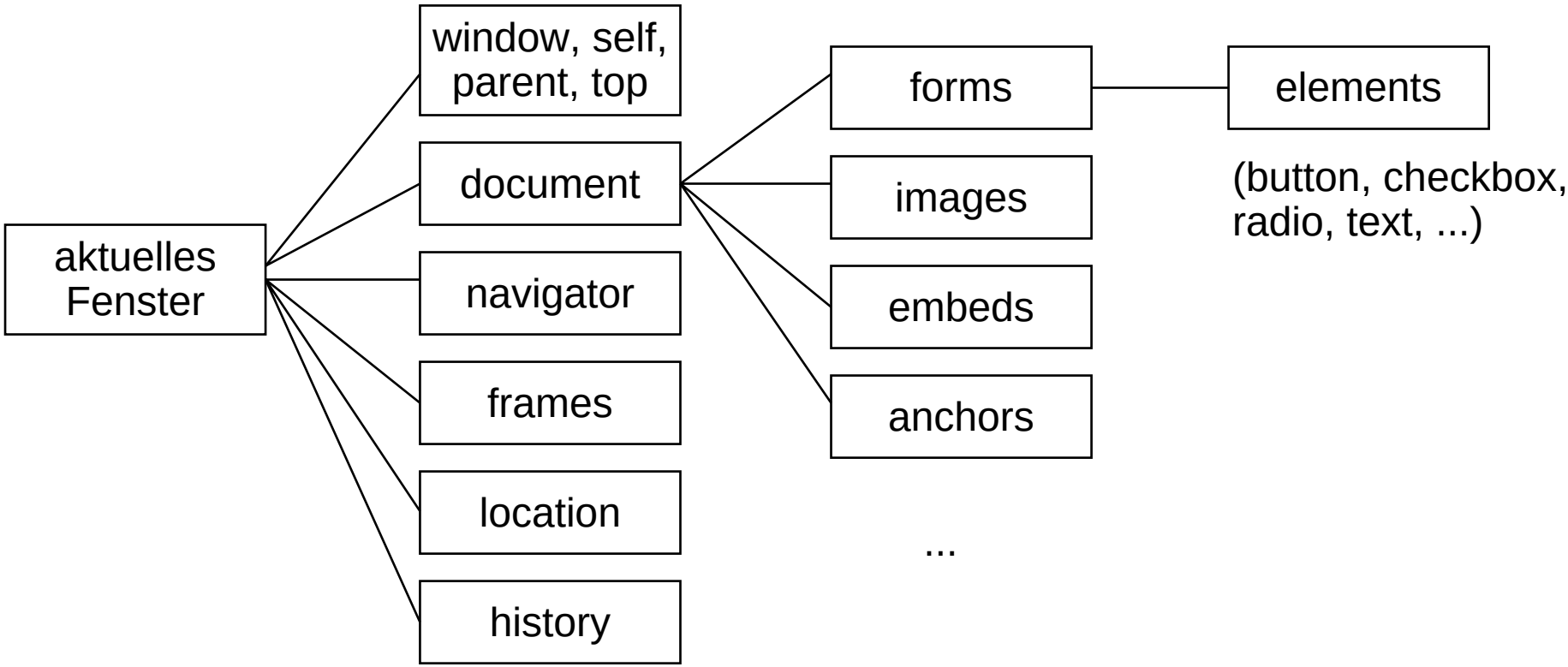
10. Skriptsprachen

Dokumentbäume für JavaScript

- Markup-Sprachen-Dokument als Baum
  - Idee ähnlich zu XML/XPath
  - Früh realisiert von Microsoft im Internet Explorer
  - Manipulation des Dokumentbaums z.B. mit JavaScript
  - Bis vor kurzem stark verschiedene Repräsentationen des Dokuments (und z.B. von Ereignissen) in verschiedenen Browsern
- Document Object Model (DOM) ist W3C-Standard
  - Siehe weiter unten
- Inkompatibilitäten verschiedener Browser:
  - Alte Netscape Browser: Keine DOM-Unterstützung
  - Alte Microsoft Browser: Proprietäre DOM-Unterstützung
  - Neueste Browser: Standardkonforme DOM-Unterstützung

Navigation im JavaScript-Dokumentbaum

- Direkter Pfad von Objekt zu Objekt:
  - Bestimmte HTML-Objekte können über Namen direkt angesprochen werden (aber nicht alle, z.B. normale Textelemente *nicht*!)
  - Häufigstes Ausgangsobjekt "document"-Objekt



Pfadnamen in JavaScript

```
function fibAlert(){
  var n = document.formular.eingabe.value;
  alert('fib('+n+') ='+fib(n));
}
...
<body> ...
     Bitte Zahlwert eingeben:
     <form name="formular">
          <input type="text" name="eingabe"
               value="0"><br>
          <input type="submit" name="knopf
               value="Berechnen">
     </form>
     <script type="text/javascript">
          document.formular.knopf.onclick=fibAlert;
     </script> ...
</body>
```

Auslesen von Kontextinformation

- Vordefinierte JavaScript-Objekte ermöglichen die dynamische Abfrage von Information

- z.B. über die Browser-Version:

```
var BrowserName = navigator.appName;
var BrowserVersion = navigator.appVersion;
```

- z.B. über die Quelldatei:

```
var Location = location;
```

Was ist DOM?

- DOM ist eine Sammlung von Hilfsmitteln für Programme, die mit Bäumen arbeiten, die XML- oder HTML-Dokumenten entsprechen
  - Level 2 in modernen Browsern realisiert
  - Level 3 (u.a. XPath-Anbindung) seit April 2004 verabschiedet
- DOM ist eine standardisierte *Programmierschnittelle* (Application Programming Interface, API)
  - Für viele verschiedene Programmiersprachen nutzbar
  - Funktionen (Name mit nachfolgenden Klammern notiert) und Eigenschaften (les- und setzbare Werte)
- Beispiele von Funktionen und Eigenschaften:

  `nodeName, nodeValue, nodeType, attributes`

  `getElementbyId()`

  `parentNode, hasChildNodes(); childNodes, firstChild, lastChild, previousSibling, nextSibling;`

  `insertBefore(), replaceChild(), removeChild(), appendChild()`

Dynamische Veränderung von Seiteninhalt

- Textknoten lassen sich über allgemeines DOM adressieren
- Mittels JavaScript können Inhalte verändert werden
- Damit wechselt der Inhalt der Webseite im Browser
- Beispiel:

```
function fibCompute(){
  var eingWert = document.formular.eingabe.value;
  var ergNode = document.getElementById("ergebnis")
   .firstChild();
  ergNode.nodeValue =
    "fib("+eingWert+") = "+fib(eingWert);
...                  }
<p id="ergebnis">
   Kein Ergebnis bisher.
</p>
...
document.formular.knopf.onclick=fibCompute;
```

Dynamische Veränderung von Stilinformation

- CSS-Attribute lassen sich durch DOM/JavaScript manipulieren
- Damit können z.B. Anzeigebestandteile ein/ausgeblendet, umformatiert und bewegt werden.
- Beispiel:

```
<form name="formular">
  <input type="text" name="eingabe" value="0"><br>
  <input type="button" name="knopf" value="Berechnen">
  <span id="hint" style="visibility:hidden;color:red;">
    Zeigt Ergebnis durch dynamische Textver&auml;nderung
  </span>
</form>...
<script type="text/javascript">
  function showHint(){
    document.getElementById("hint").
      style.visibility = "visible";
  } ...
  document.formular.knopf.onmouseover=showHint;
</script>
```