4. Internet-basierte Digitale Medien

4.1 Clientseitige Web-Skripte: JavaScript 

4.2 Document Object Model (DOM)

4.3 Serverseitige Web-Skripte: PHP

Weiterführende Literatur:

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

http://selfhtml.teamone.de

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?

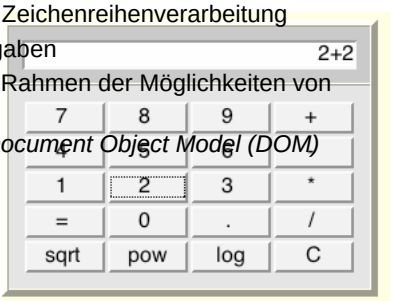
- Schlanke Programmiersprache zur integrierten Ausführung in Web-Browsern (und -Servern)
  - interpretiert
  - lokale Ausführung ohne weitere Kommunikation
  - objektbasiert (nicht echt objektorientiert, z.B. keine Klassen/Vererbung)
  - schwach typisiert
  - dynamisch gebunden
  - relativ sicher (kein Zugriff auf lokales Dateisystem und Betriebssystem)
- JavaScript hat ausser einer gewissen Syntaxähnlichkeit keine Beziehung zu Java!
- 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:
  - Eingaben in Formulare 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 Ausgabe im Rahmen der Möglichkeiten von HTML-Anzeige im Browser
  - Zugriff auf Dokument-Struktur über das *Document Object Model (DOM)* (W3C-Standard)



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*

Schwache Typisierung

- Jede Variable kann uneingeschränkt Werte eines jeden in JavaScript bekannten Datentyps annehmen:
  - Zahl (Ganzzahl, Fließkomma)
  - Zeichenreihe
  - Wahrheitswert
  - Array
  - Objekt
  - Funktion
- 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 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 über Schlüssel adressiert (analog Hashtable bzw. Map in Java)

    `a = new Array();`

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

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

"Klassen" und Objekte in JavaScript (1)

```
function sqr(x){
  return x*x;
}

// class Point
  function Point(x,y){
    this.x = x;
    this.y = y;
    this.distance = function(p){
      return Math.sqrt(sqr(p.x-this.x)+sqr(p.y-this.y))
    };
    this.display = function(){
      return "Punkt "+"
        (x = "+this.x+", y = "+this.y+")<br>"
    };
}
```


"Klassen" und Objekte in JavaScript (2)

```
p1 = new Point(1,1);
p2 = new Point(2,3);

document.writeln(p1.display());
document.writeln(p2.display());
document.writeln("Abstand: "+p1.distance(p2));
```

Anweisungen in JavaScript

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

"Vererbung" in JavaScript (1)

- Jedes Objekt enthält einen Verweis auf sein sogenanntes *Prototyp-Objekt.*
  - Werte (incl. Methoden), die nur einmal für alle Instanzen gespeichert werden sollen, kann man im Prototyp-Objekt speichern (*static* in Java).
  - Beim Erzeugen eines neuen Objekts kann man ein bestehendes Prototyp-Objekt übernehmen. Damit werden alle Eigenschaften und Methoden an das neue Objekt vererbt (können aber überschrieben werden).

```
// class Point
function Point(x,y){
  this.x = x;
  this.y = y;
}
Point.prototype.distance = function(p){
  return Math.sqrt(sqr(p.x-this.x)+sqr(p.y-this.y))
}
Point.prototype.display = function(){
  return "Punkt "+"(x = "+this.x+", y = "+this.y+")<br>"
}
```

"Vererbung" in JavaScript (2)

```
//class ColorPoint
function ColorPoint(x,y,c){
  this.x = x;
  this.y = y;
  this.color = c;
}
ColorPoint.prototype = new(Point);
ColorPoint.prototype.display = function(){
  return "Punkt "+"(x = "+this.x+", y = "+this.y+",
    color = "+this.color+")<br>"
}
...
cp1 = new ColorPoint(1,1,"red");
cp2 = new ColorPoint(2,3,"blue");
document.writeln(cp1.display());
document.writeln(cp2.display());
document.writeln("Abstand: "+cp1.distance(cp2));
```


Exkurs zu HTML: Formulare

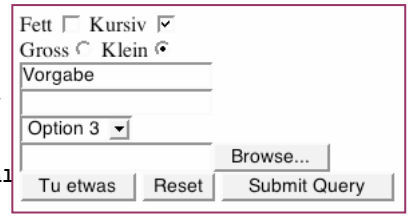
- Benutzereingabe in HTML:
  <form>-Tag
- Untergeordnete Tags:
  - **<input type=*typ* name=*name*>**
    Mögliche Typen (*typ)* (Auswahl):

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 value=*standardwert* name=*name*>**
    Liste von Optionen mit **<option value=*text*>**

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" val
   <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

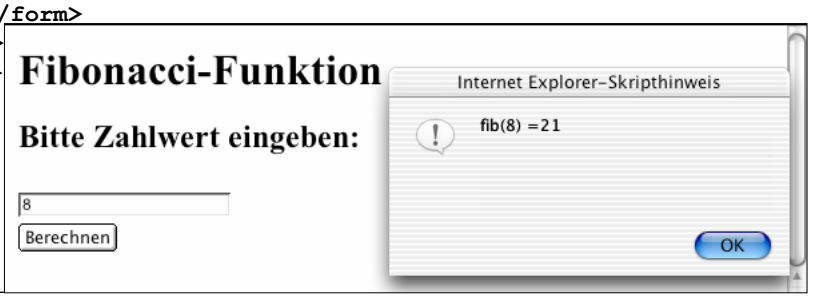
      JavaScript: **`alert(`*`String`*`)`**
  - Bestätigung:
    - » System will für eine bestimmte Entscheidung eine Bestätigung oder Ablehnung vom Benutzer erhalten. Meistens "OK"- und "Cancel"-Knopf

      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>
```



8. Skriptsprachen

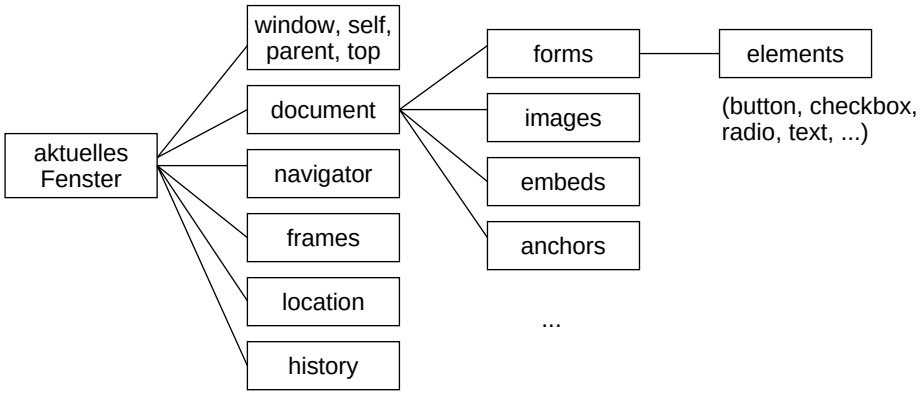
8.1 Clientseitige Web-Skripte: JavaScript
8.2 Document Object Model (DOM)
8.3 Serverseitige Web-Skripte: PHP

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
- Inkompatibilitäten verschiedener Browser:
  - Alte Netscape Browser: Keine DOM-Unterstützung
  - Alte Microsoft Browser: Proprietäre DOM-Unterstützung
  - Neueste Browser: Standardkonforme DOM-Unterstützung
- Document Object Model (DOM) ist W3C-Standard
  - Version 2.0 in modernen Browsern realisiert
  - Version 3 in Vorbereitung (u.a. XPath-Anbindung vorgesehen)

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*!)
  - Häufigster 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? (1)

DOM ist eine standardisierte Programmierschnittelle (Application Programming Interface, API) zur Inspektion und Manipulation von Objektbäumen

Definiert in der programmiersprachen-unabhängigen Schnittstellen-Definitionssprache CORBA IDL

Auszug:

```
interface Node {
  // NodeType
  const unsigned short      ELEMENT_NODE       = 1;
  const unsigned short      ATTRIBUTE_NODE     = 2;
  const unsigned short      TEXT_NODE          = 3;
  const unsigned short      CDATA_SECTION_NODE = 4;
  const unsigned short      ENTITY_REFERENCE_NODE = 5;
  const unsigned short      ENTITY_NODE        = 6;
  const unsigned short      PROCESSING_INSTRUCTION_NODE = 7;
  const unsigned short      COMMENT_NODE       = 8;
  const unsigned short      DOCUMENT_NODE      = 9;
  ...
```


Was ist DOM? (2)

```
  ...
  readonly attribute  DOMString           nodeName;
           attribute  DOMString           nodeValue;
  readonly attribute  unsigned short      nodeType;
  readonly attribute  Node                parentNode;
  readonly attribute  NodeList            childNodes;
  readonly attribute  Node                firstChild;
  readonly attribute  Node                lastChild;
  readonly attribute  Node                previousSibling;
  readonly attribute  Node                nextSibling;
  readonly attribute  NamedNodeMap        attributes;
  readonly attribute  Document            ownerDocument;
  Node insertBefore(in Node newChild, in Node refChild);
  Node replaceChild(in Node newChild, in Node oldChild);
  Node removeChild(in Node oldChild);
  Node appendChild(in Node newChild);
  boolean hasChildNodes();
  Node cloneNode(in boolean deep);
};
```

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>
```