

10. Interaktive Web-Inhalte

10.1 Clientseitige Web-Skripte: JavaScript



10.2 Dokument-Objekte und DOM

10.3 Serverseitige Web-Skripte

Weiterführende Literatur:

Stefan Koch: JavaScript: Einführung, Programmierung und Referenz – inclusive Ajax, dpunkt Verlag, 5. Auflage 2009

<http://de.selfhtml.org/>

Skriptsprachen

- Sprache zur Programmierung von Abläufen in Computersystemen
- Enge Integration mit Betriebssystem oder speziellem Anwendungssystem
- Meist interpretiert, 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, Ruby, TCL

Was ist JavaScript?

- Schlanke Programmiersprache zur integrierten Ausführung in Web-Browsern (und -Servern)
 - interpretiert
 - lokale Ausführung
 - 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 Netscape 1995 (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

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"/>
</h2>
<!-- URI -->
<h2>
  <a href="javascript:alert('Hallo');">Hallo sagen</a>
</h2>
<!-- Eventhandler -->
<h2 onClick="confirm('Halli');">
  Hier klicken...
</h2>
```

einbettung.html

JavaScript: Kommentare, Namen, Literale

- Kommentarzeilen:
 - beginnen mit `//` oder werden in `/* ... */` eingeschlossen
 - `<!--` ist ein spezieller einzeiliger Kommentar in JavaScript.
- 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
 - Spezieller einzeiliger JavaScript-Kommentar `<!--`
 - HTML-Kommentarzeichen für JavaScript auskommentieren

- Beispiel:

```
<script type="text/javascript">  
  <!--  
    document.write("Hello World!");  
  // -->  
</script>  
  
<noscript>  
  <!-- Meldung falls Skript nicht unterstützt. -->  
  <i>Bitte möglichst 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>
```

fibonacci1.html

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]

- Simulation assoziativer Arrays durch allgemeine Objekte:

- Array-Inhalt sind Schlüssel-Wert-Paare, über Schlüssel adressiert

- Simuliert wird dies durch Objekte mit Eigenschaften und Werten

- Zugriff 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": "München", "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"]
```

Ablaufstrukturen in JavaScript

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

JavaScript-Funktionen für modale Dialoge

- Dialogtypen:
 - *modal*: System wartet auf Antwort, bevor Verarbeitung fortgesetzt wird
 - » Typisches Beispiel: Öffnen-Dialog mit Dateiauswahl
 - *nicht-modal*: Dialogbearbeitung wird parallel zur normalen Arbeit fortgeführt
 - » Typisches Beispiel: Objektinspektor in Entwicklungsumgebungen
- Standardtypen von modalen Dialogen:
 - Hinweis:
 - » Sicherstellen, dass Information vom Benutzer wahrgenommen wurde
JavaScript: `alert(String)` (meist "OK"-Knopf)
 - Bestätigung:
 - » Bestätigung oder Ablehnung durch Benutzer
JavaScript: `confirm(String)` (meist "OK"- und "Cancel"-Knöpfe)
 - Abfrage:
 - » Bestimmte Eingabe vom Benutzer abrufen
JavaScript: `prompt(String, StandardwertString)`

Beispiel: Fibonacci-Programm mit Prompt

```
<body>...
  <h2>
    <script type="text/javascript">

      function fib(n) {
        ...
      }

      eing = prompt("Funktionsparameter", "0");
      document.writeln("fib("+eing+") = "+fib(eing)+"<br>");
    </script>
  </h2>
</body>
```

fibonacci_prompt.html



fibonacci_prompt.html

Exkurs zu HTML: Formulare

- Benutzereingabe in HTML:

- `<form>`-Element

- Untergeordnete Elemente:

- `<input type=typ name=name>`

- Mögliche Typen (*typ*) (Auswahl):

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

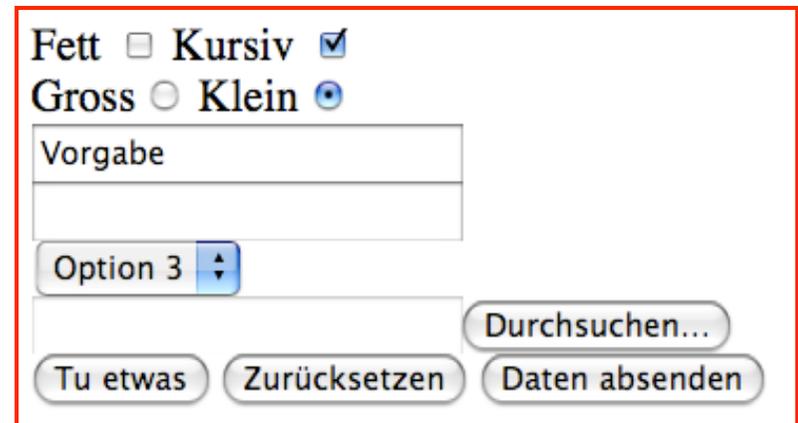
- `<select name=name>`

- Liste von Optionen: Untergeordnete Elemente vom Typ `<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>
```



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. Interaktive Web-Inhalte

10.1 Clientseitige Web-Skripte: JavaScript

10.2 Dokument-Objekte und DOM



10.3 Serverseitige Web-Skripte

Weiterführende Literatur:

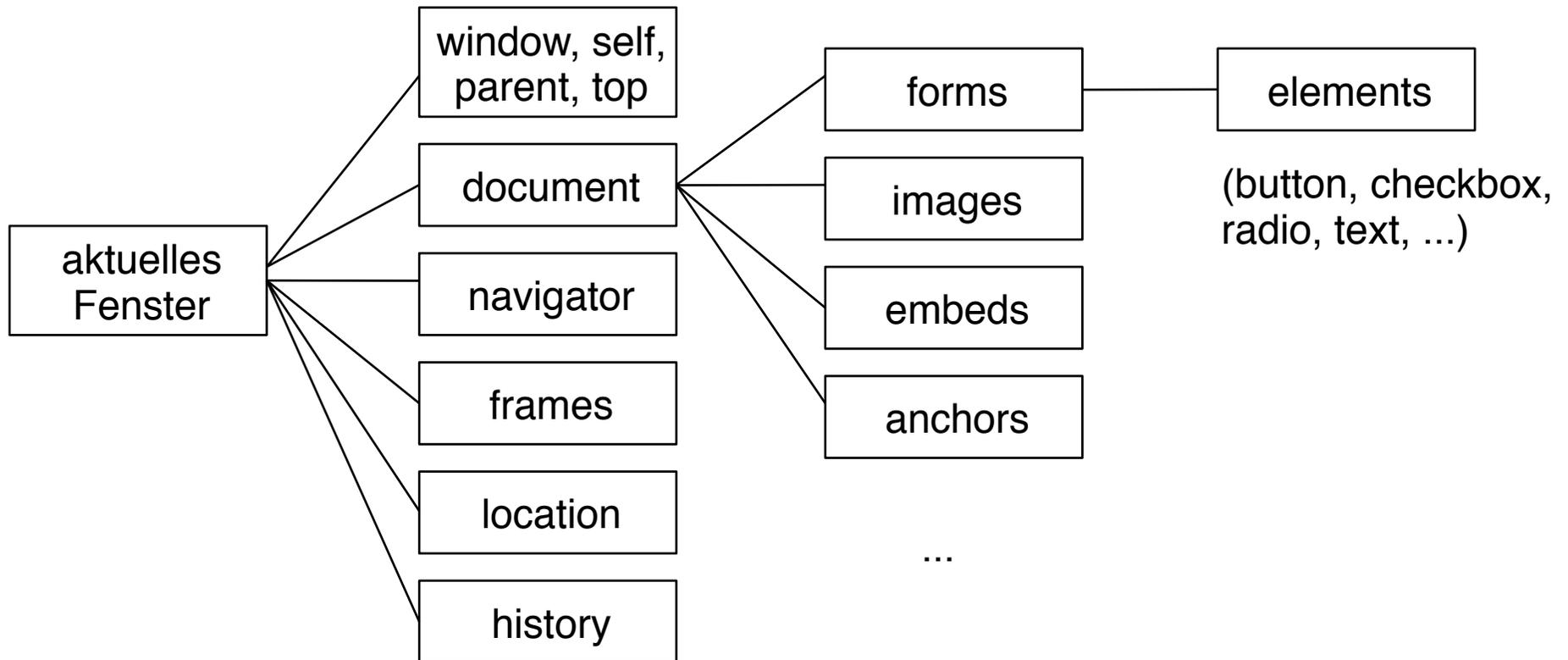
Jeremy Keith, Jeffrey Sambells: DOM Scripting: Web Design with JavaScript and the Document Object Model, Friends of ed, revised edition 2010

Dokumentbäume für JavaScript

- Markup-Sprachen-Dokument als Baum
 - Früh realisiert von Microsoft im Internet Explorer
 - Manipulation des Dokumentbaums z.B. mit JavaScript
 - Repräsentationen des Dokuments (und z.B. von Ereignissen) grundsätzlich Browser-abhängig
 - JavaScript enthält Standard-Objekte zum Zugriff auf Dokumentteile
- Document Object Model (DOM) ist W3C-Standard für Dokumentbäume
 - Siehe später

Vordefinierte JavaScript-Objekte

- Direkter Pfad von Objekt zu Objekt:
 - Häufigstes Ausgangsobjekt "document"-Objekt
 - Objekte stellen Array von Unterobjekten zur Verfügung
 - Unterste Unterobjekte sind HTML-Elemente



Eigenschaften von HTML-Objekten

- Jedes HTML-Objekt hat Eigenschaften (lokale Variable)
 - Jedes Attribut des HTML-Elements ist eine Eigenschaft
 - Universaleigenschaften (z.B. className, id)

- Notation:

objekt . Eigenschaft

- Beispiel:

```
<input type="text" name="eingabe" value="0">
```

sei repräsentiert als JavaScript-Objekt `o`

- Zugriff auf den Wert des Textfeldes:

```
o.value
```

Beispiel zu vordefinierten JavaScript-Objekten

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

Alternative Schreibweise zum gleichen Beispiel

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

fibonacci3a.html

Kurzschreibweise

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

Direkte Adressierung über Element-Name
(kurz aber stilistisch nicht sehr schön)

fibonacci4.html

Auslesen von Kontextinformation

- Vordefinierte JavaScript-Objekte ermöglichen die dynamische Abfrage von Information

- z.B. über die Browser-Version:

```
var userAgent = navigator.userAgent;  
var browserName = navigator.appName;  
var browserCodeName = navigator.appCodeName;  
var browserVersion = navigator.appVersion;  
var platform = navigator.platform;
```

- Hinweis: Moderne Browsernamen (Firefox, Safari etc.) sind in der Regel als Teilzeichenreihe in *userAgent* und/oder *appVersion* codiert.

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

fibonacci5.html

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

fibonacci5.html

10. Interaktive Web-Inhalte

10.1 Clientseitige Web-Skripte: JavaScript

10.2 Dokument-Objekte und DOM

10.3 Serverseitige Web-Skripte

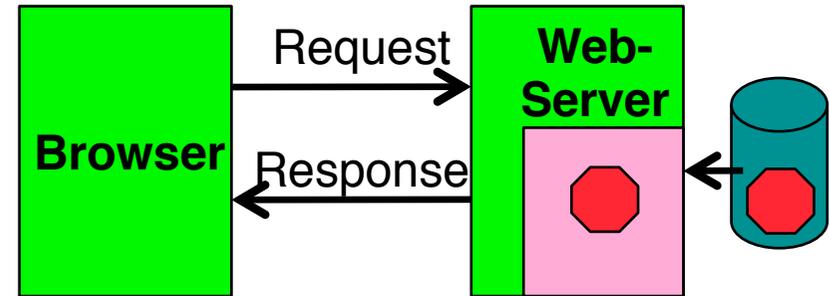
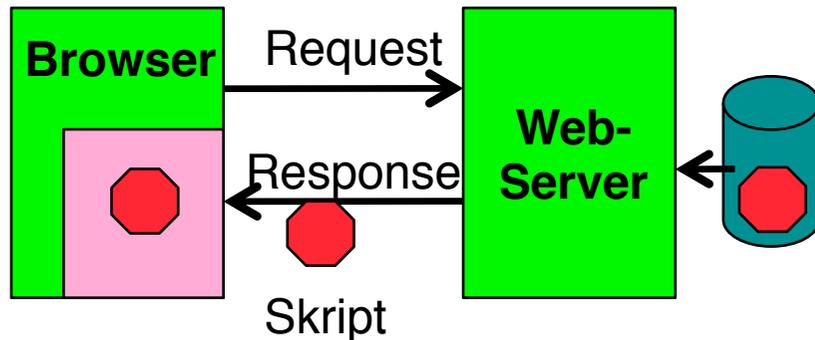


Weiterführende Literatur:

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

<http://de.selfhtml.org/>

Serverseitige vs. clientseitige Dynamik



- Clientseitige Dynamik:

- Browser enthält Ausführungsmaschine für Skripte
- Skript ist Teil der Antwort vom Server
- Web-Server muss Skriptsprache nicht kennen
- Beispiel: JavaScript

- Serverseitige Dynamik:

- Web-Server enthält Ausführungsmaschine für Skripte
- Skript wird vor Beantwortung der Anfrage ausgeführt und liefert HTML-Text
- Browser muss Skriptsprache nicht kennen
- Beispiel: PHP

Technologien für serverseitige Dynamik

- Common Gateway Interface (CGI):
 - Ermöglicht Aufruf beliebiger Programme beim Server (z.B. in C)
 - Programm erzeugt (schreibt) HTML-Textdatei (Response)
 - Häufig Skriptsprachen benutzt (z.B. Perl, Python)
 - Manchmal spezielle Bibliotheken für Webseiten verfügbar
- Spezielle Server-Skriptsprachen (z.B. PHP):
 - Entworfen für Einbettung in HTML
 - Plug-Ins für gängige Web-Server-Software
 - geeignet für "mittelgroße" dynamische Anwendungen
- Programmiersprachen-Einbettung in Web-Server:
 - z.B. Java *Servlets*, geschrieben in Java
 - Aufgerufen vom Server über standardisiertes API
 - Generierung von Servlets aus Skript-ähnlichen Sprachen z.B. Java Server Pages (JSP)
 - geeignet für "große" dynamische Web-Anwendungen

Beispiel: Server-Skriptsprache PHP

- PHP:
 - Personal Home Page Toolkit
 - PHP Hypertext Preprocessor
- OpenSource Entwicklung:
 - siehe www.php.net
 - lizenzfrei benutzbar
- Syntax an C angelehnt, aber mehrere Syntax-Varianten unterstützt
- Einfache Kernsprache, umfangreiche Funktionsbibliothek
 - über 500 Funktionen!
 - etwas unübersichtlich
 - spezialisiert auf Aufgaben der Webseiten-Programmierung

Voraussetzungen für praktische Experimente

- Auch bei lokalen (Ein-Rechner-)Experimenten
 - Installation eines Web-Servers
 - » OpenSource: *Apache*
 - » Microsoft *Internet Information Server*
 - Aufruf der HTML-Dateien über Web-Server (<http://...>)
- Bereitstellung und Installation der PHP-Software als Plug-In für den verwendeten Web-Server
- In den meisten praktischen Fällen: Installation eines relationalen Datenbanksystems (z.B. MySQL)
- Insider-Kürzel für bestimmte Konfigurationen (Beispiele):
 - LAMP: Linux, Apache, MySQL, PHP
 - WIMP: Windows, Internet Information Server, MySQL, PHP
 - MOXAMP: MacOS X, Apache, MySQL, PHP (hier verwendet)

Beispiel: "Hello World" in PHP und JavaScript

```
<html>
<head><TITLE>Hello World mit JavaScript</TITLE></head>
<body>
  <h1>
    <script type="text/javascript">
      document.write("Hello World!");
    </script>
  </h1>
</body>
</html>
```

JavaScript

```
<html>
<head><title>Hello World mit PHP</title></head>

<body>
  <h1>
    <?php
      echo "Hello World!";
    ?>
  </h1>
</body>
</html>
```

PHP

Einbettung von PHP in HTML

- XML-Stil (hier verwendet):
 - Analog zu *Processing Instructions* von XML
 - `<?php PHP-Text ?>`
- SGML-Stil:
 - Kurze und weit verbreitete "Urform"
 - Nicht empfehlenswert, da PHP nur default-Annahme
 - `<? PHP-Text ?>`
- HTML-Stil:
 - Analog zur JavaScript-Einbettung
 - `<script language="php"> PHP-Text </script>`

(Lästige) Details: Syntaktische Unterschiede

- Generell stärkere Anlehnung an Shell-Skriptsprachen
 - Variablen beginnen immer mit "\$"
 - Viele UNIX-Kommandos direkt verfügbar, z.B.

```
echo "Beispiel";
```


(statt in JavaScript: `document.write("Beispiel");`)
- Verschiedene Varianten für Steueranweisungen, z.B.:

```
if (bedingung1) anw1 elseif (bedingung2) anw2 else anw3;  
if (bedingung1): anwfolge1 elseif (bedingung2): anwfolge2  
else: anwfolge3 endif;
```
- Schwach typisiert, aber geringfügig anderes Typsystem als JavaScript
- Arrays einschließlich assoziativer Arrays, aber etwas andere Syntax und Bibliothek als in JavaScript
- PHP ist weitgehend objektorientiert, kennt Klassen und Vererbung in Java-Syntax.

Server-Skripte und Formulare

- Benutzereingaben aus Formularen
 - Müssen zuerst zum Server übertragen werden
 - Werden dann im Server-Skript ausgewertet
 - Werden dann lokal angezeigt, indem eine neue HTML-Seite generiert wird
- HTML: Attribut **action** beim Formular-Tag **<form>**
 - Spezifiziert das zur Verarbeitung der Eingabe benutzte Server-Dokument
 - Typische Beispiele:
 - » PHP-Skript
 - » Email-Versand (`action=mailto:xyz@abc.com`)
 - » HTML-Seite mit eingebetteten Skripten
 - Einfacher Spezialfall:
 - » Aufruf der aktuellen Seite (Neuladen)

Fibonacci-Funktion mit PHP: Eingabeseite mit Aufruf von PHP-Skript

```
<body>
  <h1>
    Fibonacci-Funktion (Eingabe)
  </h1>
  <h2>
    Bitte Zahlwert f&uuml;r Berechnung eingeben:
    <form name="formular" action="fibonacci2b.php">
      <input type="text" name="eingabe"
        value="0"><br>
      <input type="submit" value="Berechnen">
    </form>
  </h2>
</body>
</html>
```

Datei fibonacci2a.html

Fibonacci-Funktion mit PHP (Version 2): Ergebnisseite

```
<body>
  <h1>
    Fibonacci-Funktion (Ergebnis)
  </h1>
  <h2>
    <?php
      $eingabe = $_REQUEST['eingabe'];
      function fib($n)
        { Fibonacci berechnen };
      echo "fib($eingabe) = ";
      echo fib($eingabe);
      echo "<br>";
    ?>
    <br>
    <a href="fibonacci2a.html">Neue Berechnung</
  a>
</h2>
</body>
```

Datei fibonacci2b.php

GET- und POST-Methode in HTTP

- Das Hypertext Transfer Protocol (HTTP) unterstützt zwei Methoden, Parameterwerte an aufgerufene Dokumente zu übergeben
- GET-Methode:
 - Variablenwerte werden als Bestandteil der URL codiert und übergeben:
`http://host.dom/pfad/fibonacci2.php?eingabe=12`
 - Damit können Parameterangaben auch durch Eintippen der URL gemacht werden (ohne Formular)
 - Geeignet für einfache Abfragen
- POST-Methode:
 - Variablenwerte werden nicht in der URL codiert
 - Webserver wartet auf anschließende Übertragung der Variablenwerte (Einlesen vom Standard-Eingabekanal)
 - (Etwas) schwerer von außen zu "manipulieren"
- HTML: Attribut `method` beim Formular-Tag `<form>`
 - `method="get"` (default!) oder `method="post"`

Server-Skripte vs. Client-Skripte

Client-Skripte

Schnelle Reaktion
Funktion auch ohne Netzanbindung
Unabhängigkeit von Server-Software

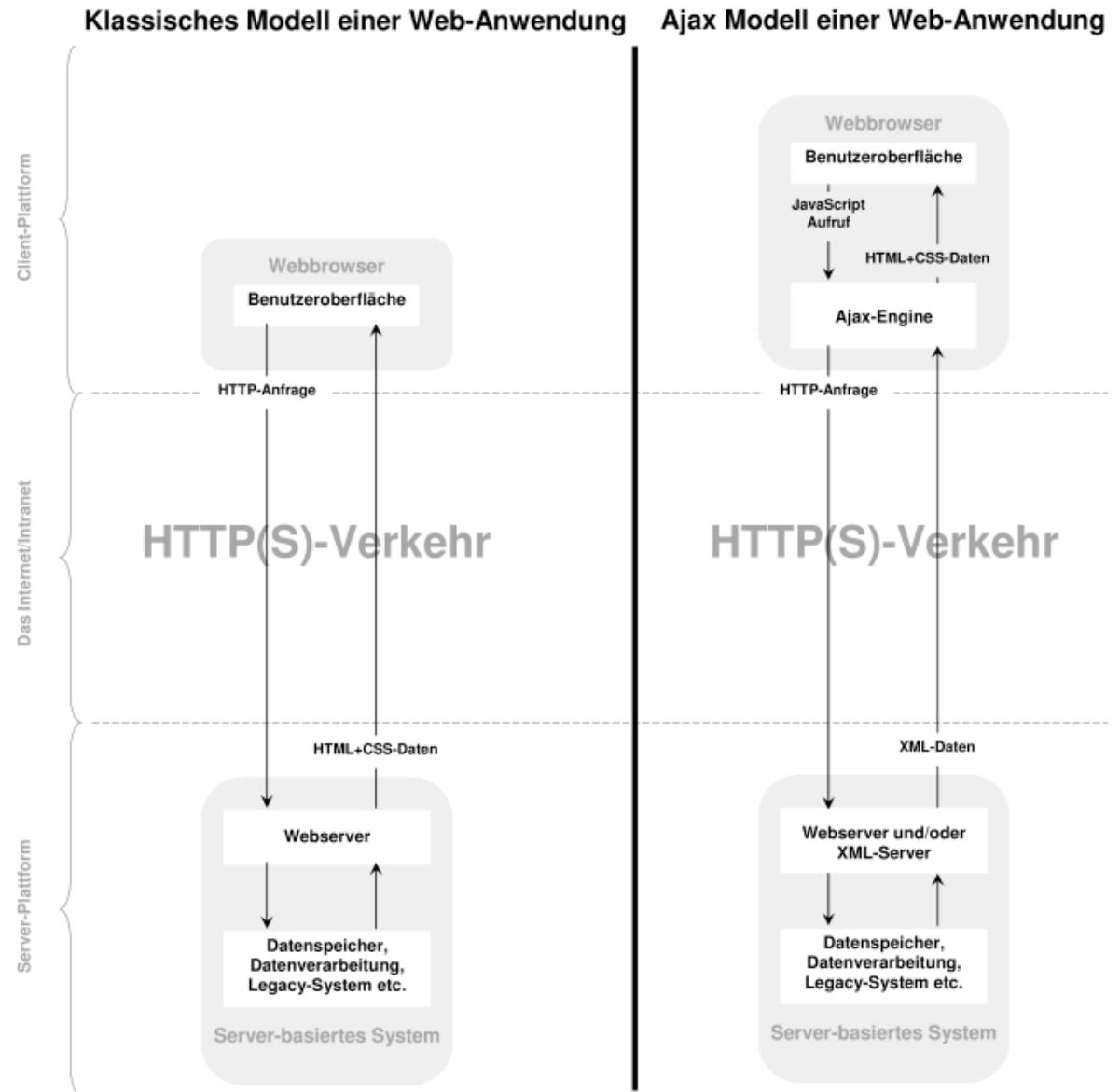
Berechnung von Seiteninhalt aus
Benutzereingaben und anderen
äußeren Umständen

Server-Skripte

Datenhaltung auf Server
Zugriff auf zentrale Ressourcen (z.B. zur Weiterverarbeitung)
Unabhängigkeit von Browser-Software

AJAX und Comet

- AJAX: Asynchronous JavaScript and XML
- Konzept der asynchronen Datenübertragung zwischen Server und Browser
- HTTP-Anfrage wird an Server geschickt ohne Seite neu zu laden
 - (XML-)Antwort wird in JavaScript ausgewertet
- Comet: "reverse AJAX"
 - Push-Technologie, Server sendet Information asynchron zum Browser



Bildquelle: Wikipedia

WebSockets

- Programmier-Schnittstelle (API), vom World Wide Web Consortium (W3C) definiert
- Ermöglicht es einer Web-Anwendung, mit einem beliebigen Host einen *full-duplex* Kommunikationskanal zu unterhalten
- Noch in Entwicklung
 - JavaScript-Version des API existiert
 - Integriert in den HTML5 Standard
 - Sicherheitsbedenken bremsen den praktischen Einsatz derzeit
- Prinzipiell ein Weg, Ineffizienzen und unnötige Verkehrslast derzeitiger Ajax-ähnlicher Anwendungen zu reduzieren