Digitale Medien

Blockpraktikum

Informationsvisualisierung mit Processing

- **Was?**
  - Praktischer Einstieg in die Informationsvisualisierung anhand des Java Frameworks „Processing“
  - Umsetzung eines kleinen Visualisierungsprojekts in Zweiergruppen
  - Anrechenbar für „Projektkompetenz Multimedia“ (3 ECTS)
- **Wann?**
  - 1. 3. – 9. 3. 2012 (ohne Wochenende ;)
- **Fragen?**
  - Mail an buschek@cip.ifi.lmu.de, preisj@cip.ifi.lmu.de

- **Anmeldung und weitere Infos:**
  - http://www.medien.ifi.lmu.de/ivp

10. Interaktive Web-Inhalte

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

Marjin Haverbeke: Die Kunst der JavaScript Programmierung. dpunkt Verlag, 1. Auflage 2012

http://de.selfhtml.org/

Skriptsprachen

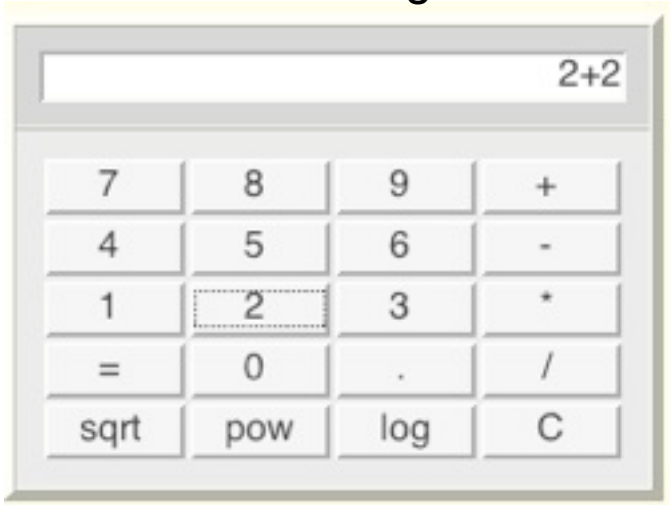
- Sprache zur Programmierung von Abläufen in Computersystemen
- Enge Integration mit Betriebssystem oder speziellem Anwendungssystem
- Meist interpretiert, leicht zur Laufzeit zu definieren und zu ändern
- Moderne Skriptsprachen durchaus Alternative zu Programmiersprachen
- Beispiele:
  - Betriebssystem-Skripte: 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

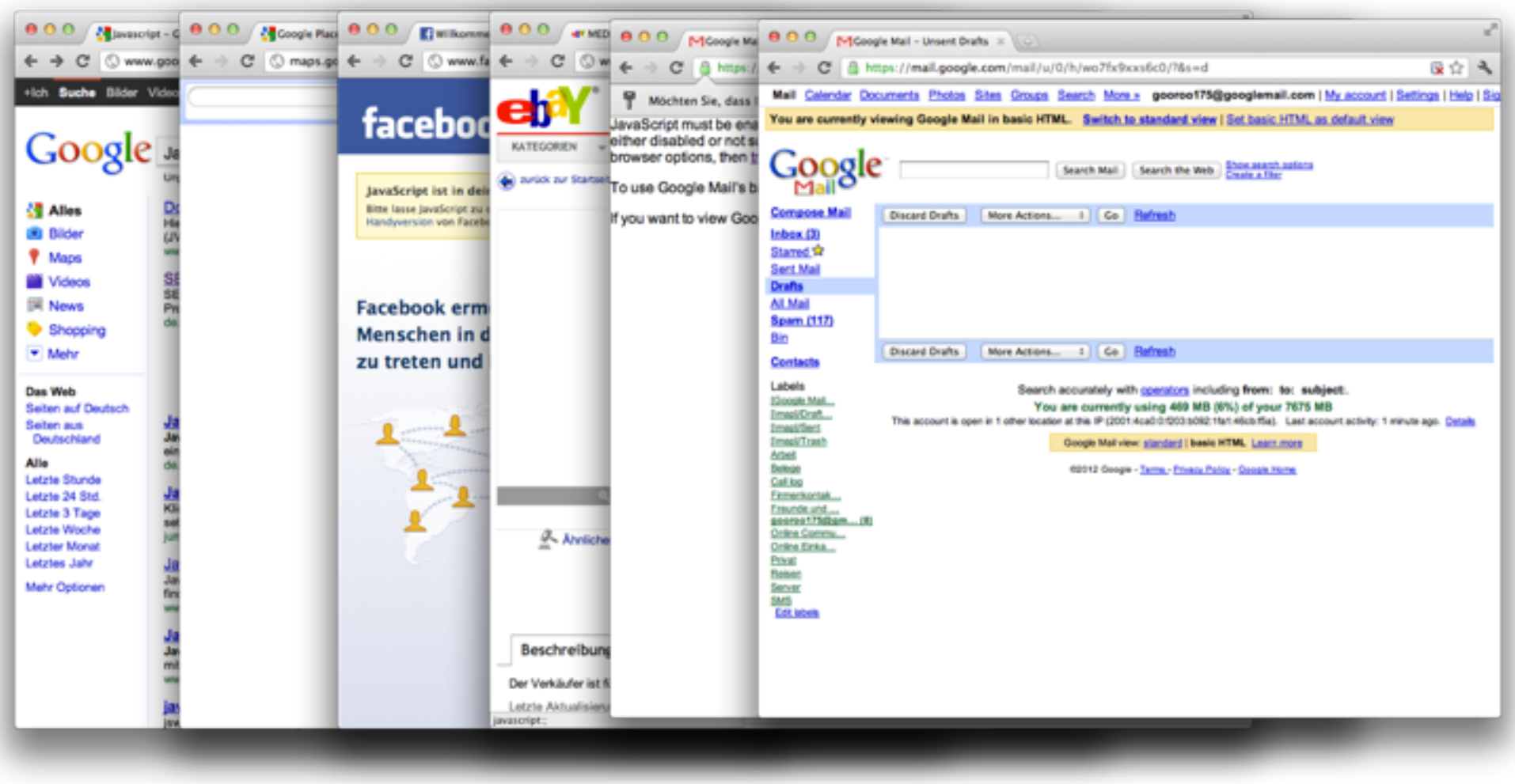
JavaScript: Funktionsumfang und Anwendungsbereich

- 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)*



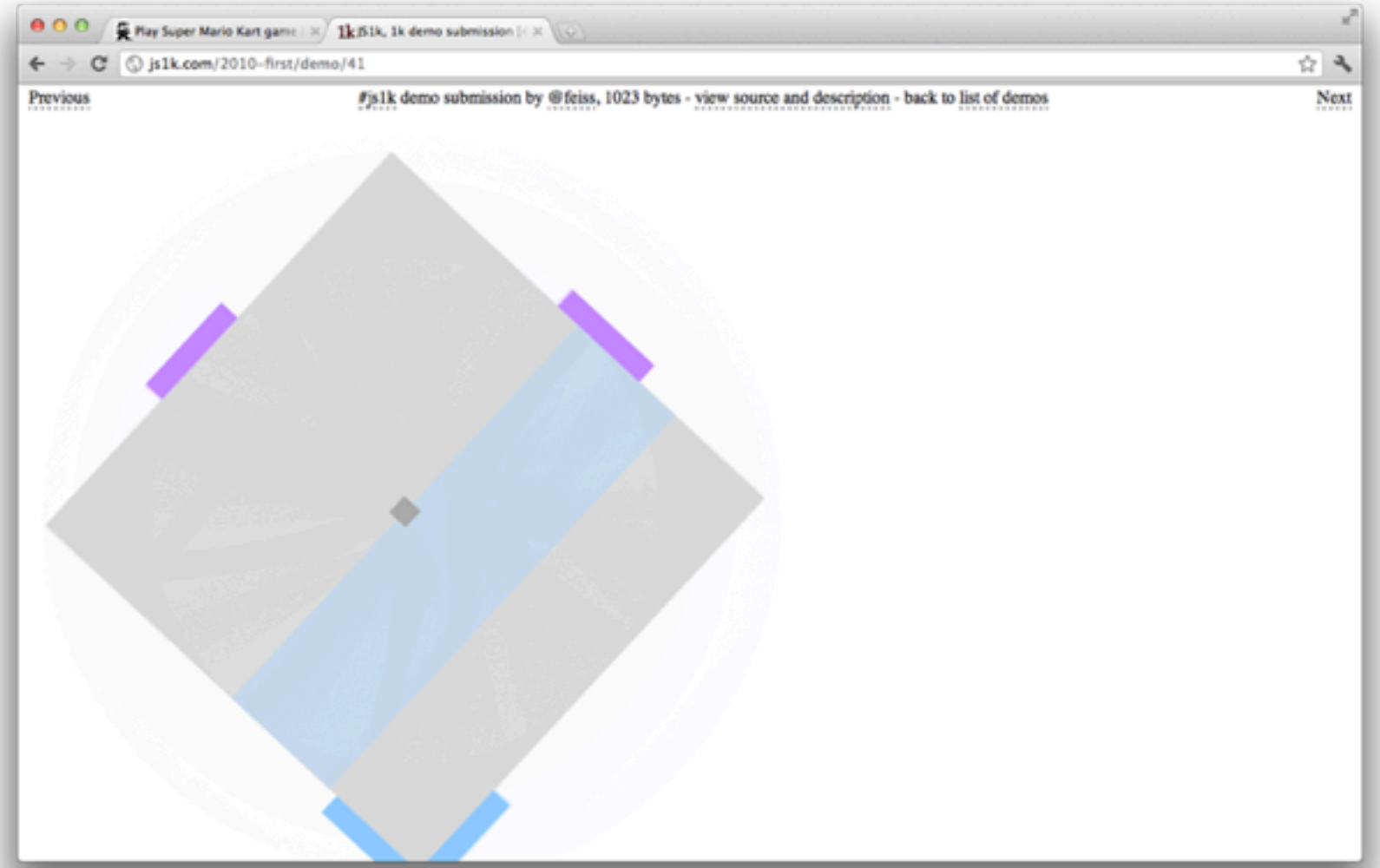
http://de.selfhtml.org/javascript/beispiele/anzeige/taschenrechner.htm

Die Welt ohne JavaScript



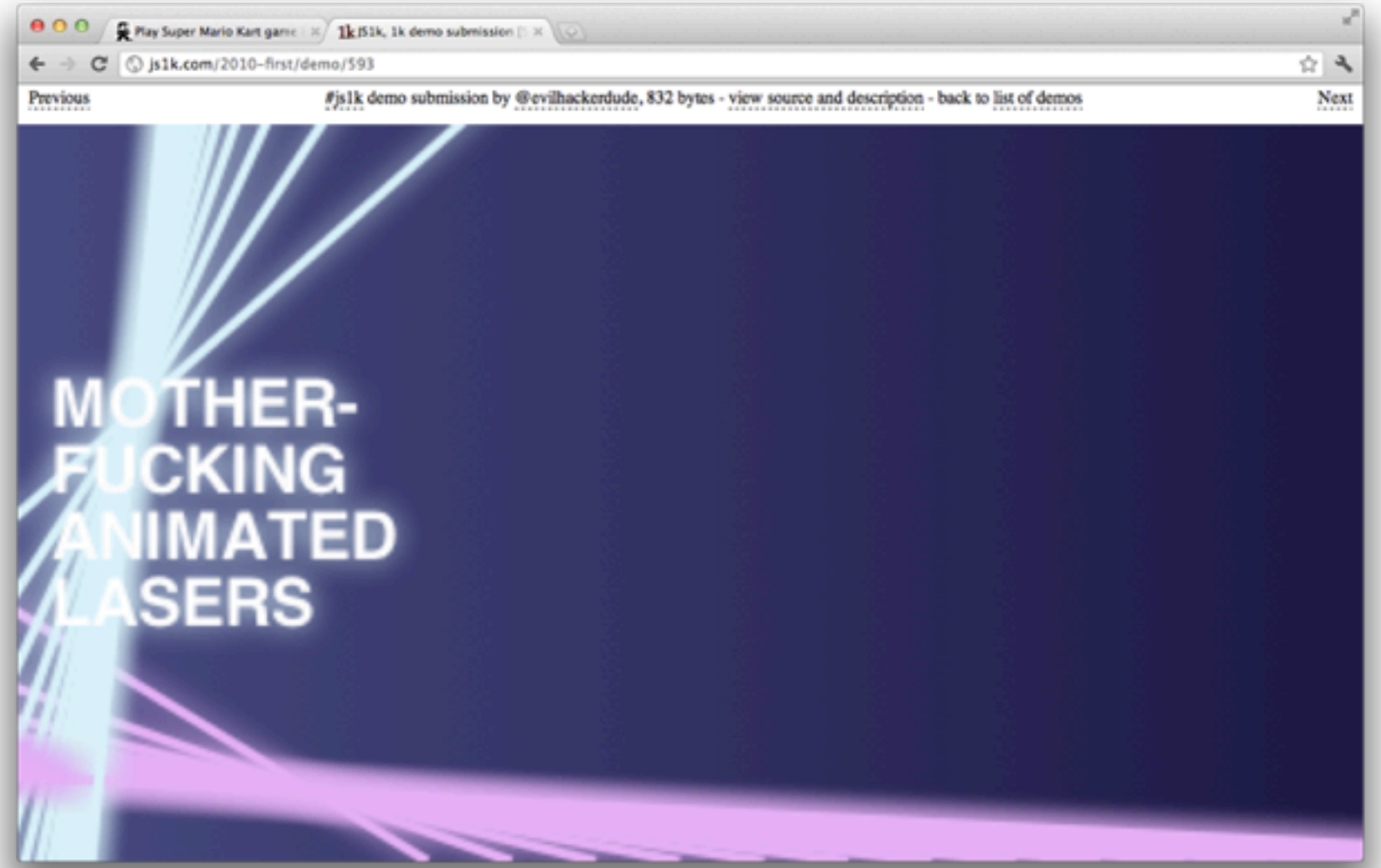
JavaScript: Beispiel 1

http://js1k.com/2010-first/demo/41



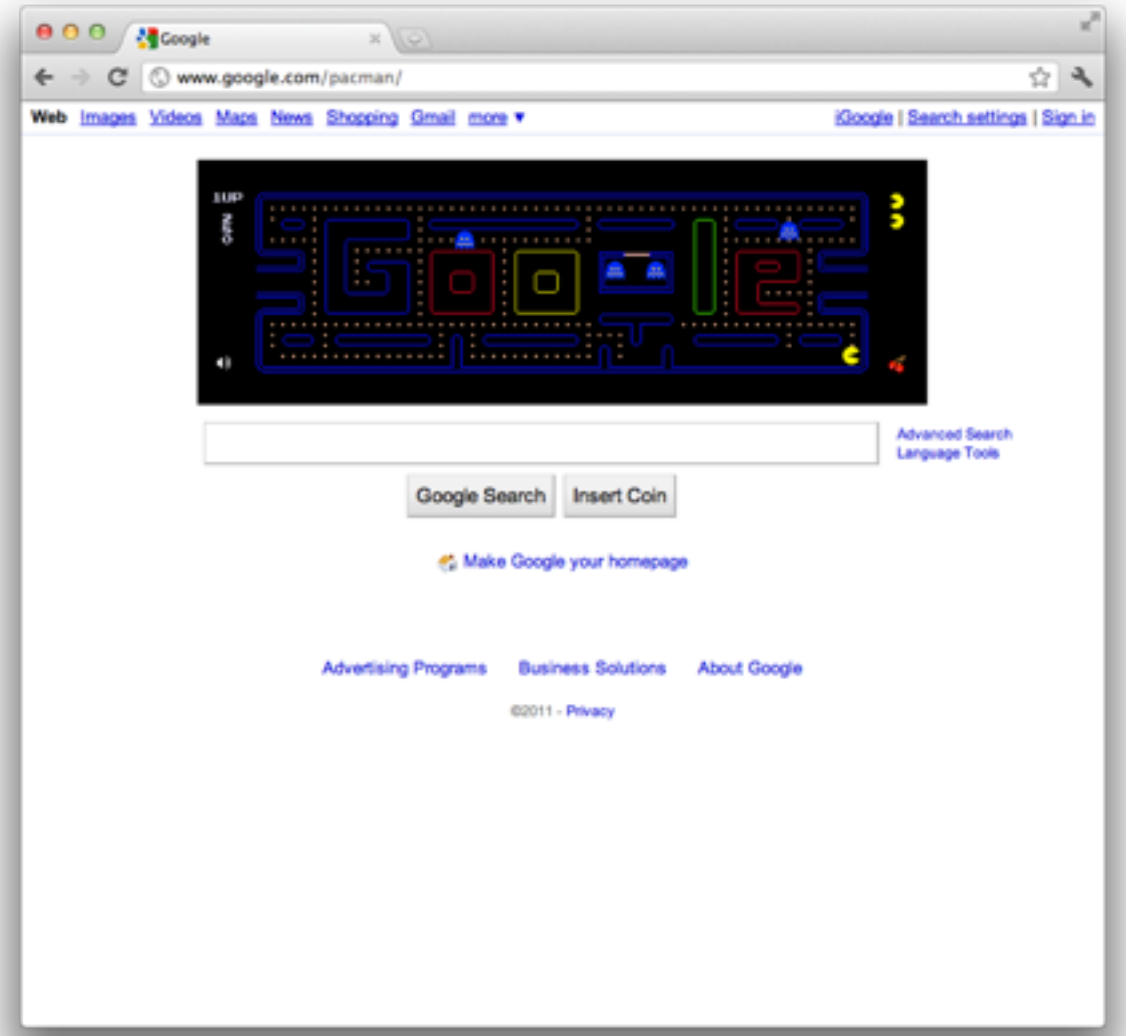
JavaScript: Beispiel 2

http://js1k.com/2010-first/demo/593



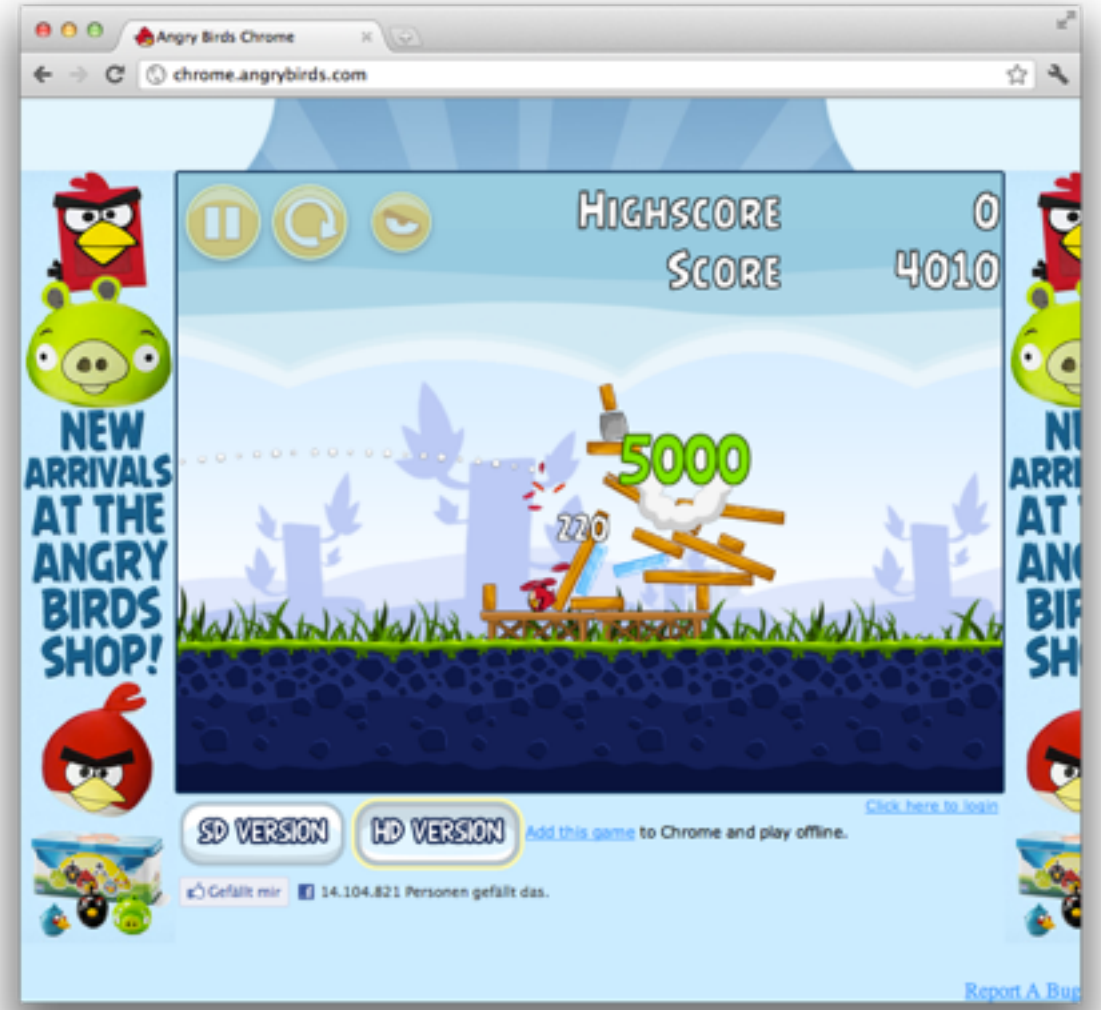
JavaScript: Beispiel 3

http://www.google.com/pacman/
http://www.google.com/logos/js/pacman10-hp.5.js



JavaScript: Beispiel 4 (JS + HTML 5)

http://chrome.angrybirds.com/



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>
```

Code wird bei erreichen ausgeführt. Auch Funktionsdefinitionen.

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:

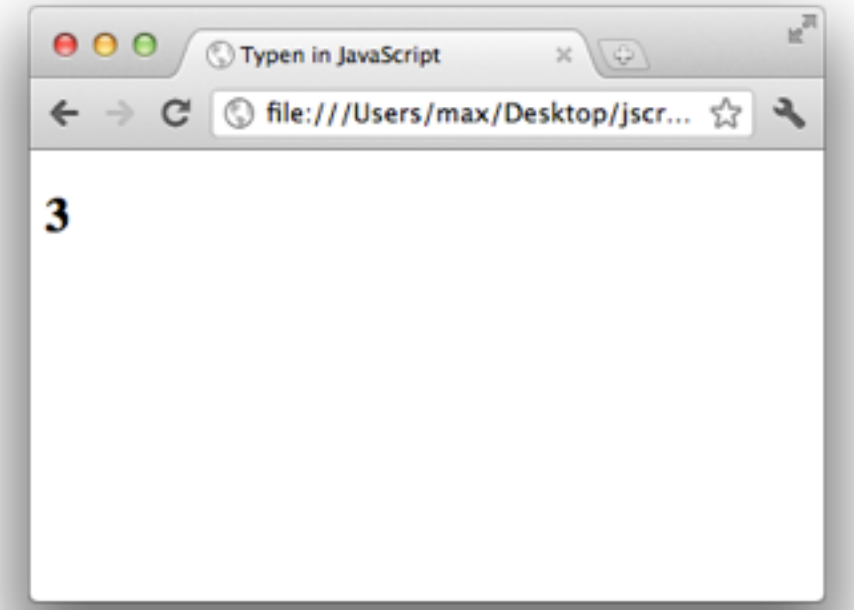
```
<script type="text/javascript">
  <!--
     document.write("Hello World!");
  // -->
</script>

<noscript>
  <!-- Meldung falls Skript nicht unterstützt. -->
     <i>Bitte 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`**

Schwache Typisierung (Problem)

```
<script type="text/javascript">
    var i=0;
    i++; // i=1;
    i++; // i=2;
    I=1; // i=1?
    i++; // i=2?
    document.writeln(i);
</script>
```



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 (Hashes) 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";
a = {"x":"X", "y":"Y"};
```

Lesen: `a["x"]` `a.x`

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"]`**

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

fibonacci_prompt.html

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

      function fib(n){
        ...
      }

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

Exkurs zu HTML: Formulare

- Benutzereingabe in HTML:
  `<form>`–Element
- Untergeordnete Elemente:
  - **`<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: Untergeordnete Elemente vom Typ **`<option>`**
    `<option selected>` bestimmt "vorselektierten" Standardwert

Beispiel: HTML-Formular

forms.html

```
<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="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="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

fibonacci2.html

```
<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>
```

JavaScript im (un)sinvollen Einsatz

http://gifsound.com/?gif=http%3A%2F%2Fdyehlah.files.wordpress.com%2F2011%2F03%2Ftumblr_ldnle8ma1s1qby2pf.gif&sound=http%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3DQH2-TGUlwu4&start=32

10. Interaktive Web-Inhalte

Weiterführende Literatur:

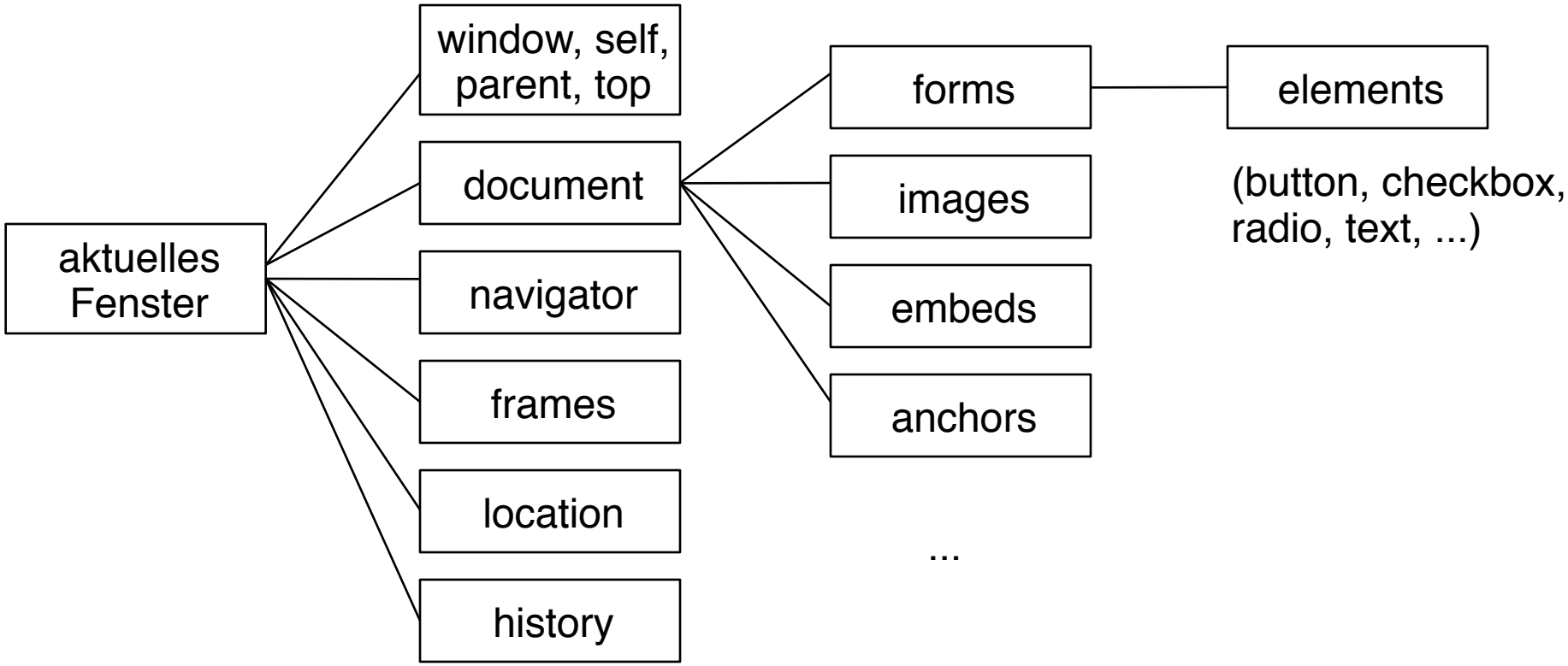
Jeremy Keith, Jeffrey Sambells: DOM Scripting: Web Design with JavaScript and the Document Object Model, Friends of ed, revised edition 2010

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`

Achtung: Nicht für alle gültig.
nicht „o.class“ sondern
„o.className“
Ansonsten immert
o.getAttribute(„class“);

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>
```

fibonacci3.html

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;
```

navigator.html

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 Textver&auml;nderung
  </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 

10.4. Beyond JavaScript: Prototype, jQuery, script.aculo.us und HTML5

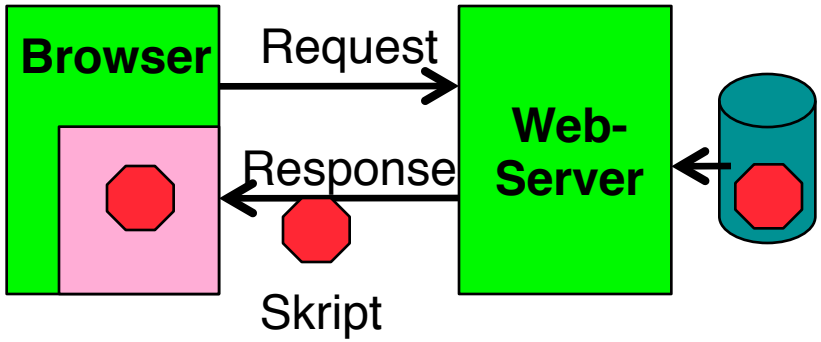
10.5. Und was ist das?

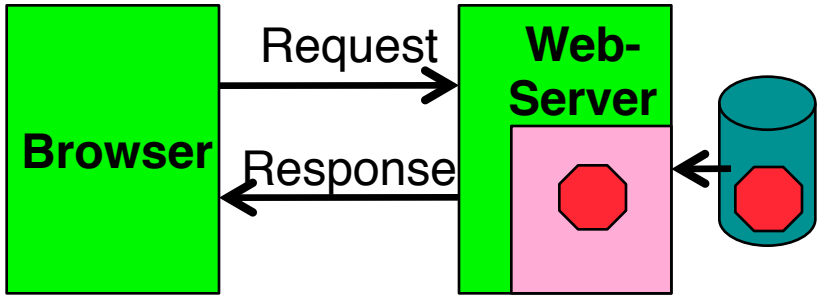
Weiterführende Literatur:

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:
  - **P**ersonal **H**ome **P**age Toolkit
  - **P**HP **H**ypertext **P**reprocessor
- 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

PHP

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

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

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>
```

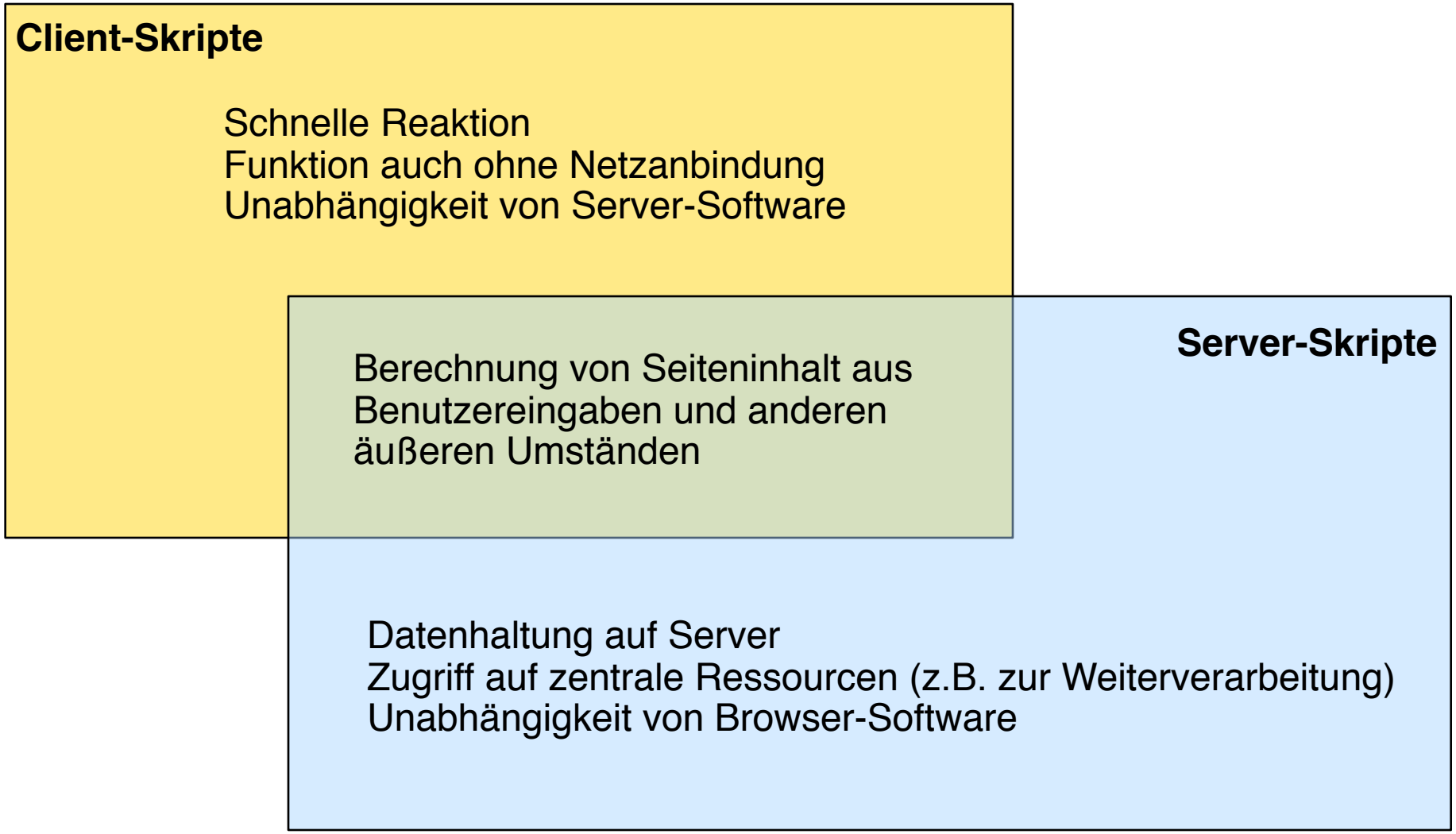
Datei fibonacci2b.php

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

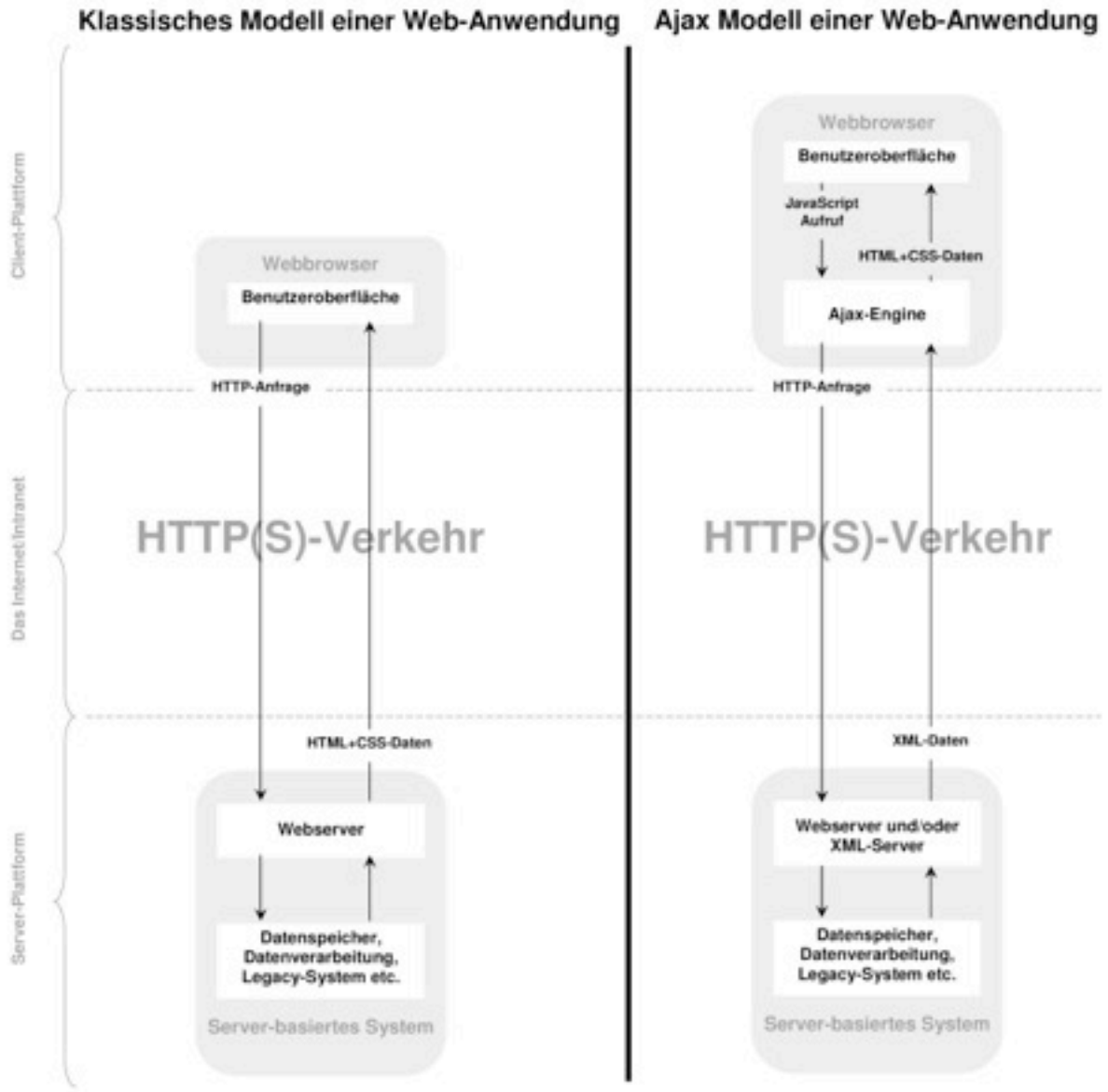


Im Direkten Vergleich

Client-Skripte	Server-Skripte
weniger CPU-Last für Server	weniger CPU-Last für Client
Eingabenvalidierung nicht gewährleistet	Eingabenvalidierung gewährleistet
Direktere Reaktionszeit	Warten auf Serverantwort erforderlich
Performance schlecht kontrollierbar	Performance kontrollierbar
Code Sichtbar	Code nicht sichtbar
Obfuscation notwendig	Obfuscation nicht nowendig
Arbeitsdaten müssen alle übertragen werden	Arbeitsdaten werden erst reduziert und dann übertragen

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

10. Interaktive Web-Inhalte

10.1 Clientseitige Web-Skripte: JavaScript

10.2 Dokument-Objekte und DOM

10.3 Serverseitige Web-Skripte

10.4. Beyond JavaScript: Prototype, jQuery, script.aculo.us und HTML5 

10.5. Und was ist das?

Weiterführende Literatur:

http://www.w3.org/TR/html5/

JS und DOM sind schön, aber....

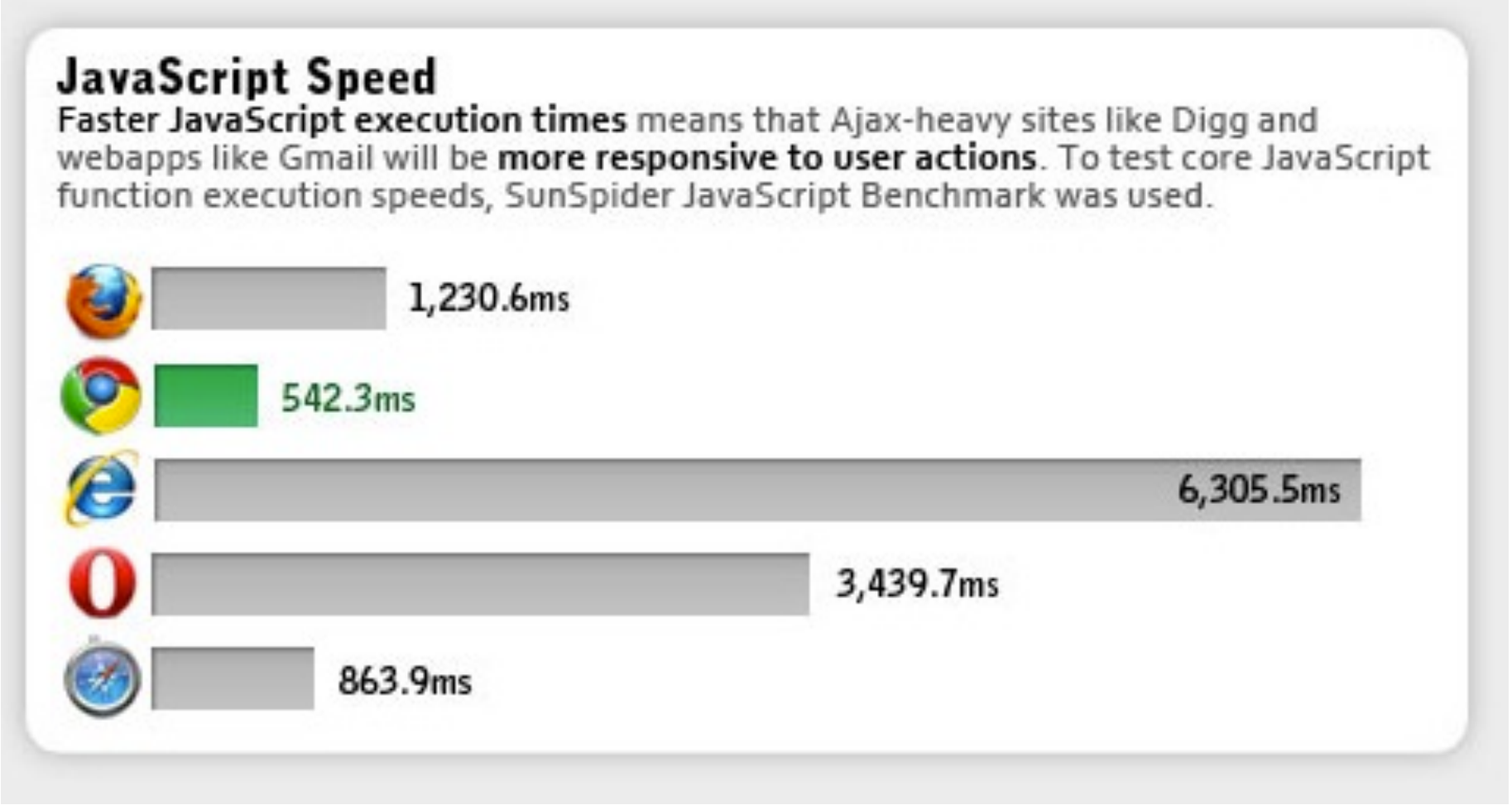
- ...JavaScript ist langsam!
  - Andere Programmiersprache?
  - Statt dessen: Schnellere Interpreter
  - Abhilfe: **JavaScript Benchmarks**
- ...es gibt immer noch Browserinkompatibilitäten
  - Code muss Unterscheidungen je nach Browser machen
  - Beispiel Fenstergröße:
    - » IE: `var w = document.documentElement.clientWidth;`
    - » Firefox: `var w= window.innerWidth;`
  - Abhilfe: Frameworks wie **jQuery** oder **prototype**
- ...nicht allumfassend
  - Dateidialoge schauen immer gleich aus
  - Audio/Video nur schwer möglich
  - Zeichnen von Formen und Grafiken nicht möglich
  - Abhilfe: **HTML5**

JavaScript Benchmarks

Ein **Benchmark** misst die Ausführungszeit von festgelegten Operationen und macht so verschiedene Systeme vergleichbar.

- Wettrüsten der Browserhersteller
- Verschiedene Bereiche von JavaScript
  - Mathematische Funktionen
  - Datumsfunktionen
  - Stringfunktionen
  - Bitweise Operationen
- Verschiedene Benchmarks versuchen dies alles zu testen und die Zeit zu messen
  - SunSpider: http://www.webkit.org/perf/sunspider/sunspider.html
  - Peacekeeper: http://peacekeeper.futuremark.com/
  - Google V8: http://v8.googlecode.com/svn/data/benchmarks/v6/run.html

JavaScript Benchmarks



http://sixrevisions.com/infographs/browser-performance/

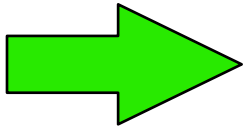
JavaScript Zusatzframeworks

- JavaScript nicht in allen Browser eindeutig
- Außerdem manche häufig verwendete Funktionen umständlich zu programmieren
- z.B.: `document.getElementById("element").value`
- jQuery oder prototype schaffen Abhilfe
- Beide funktionieren ähnlich:
  - Erweiterung der Standard-Java-Befehlspalette

jQuery

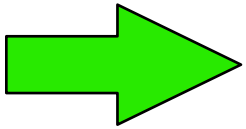
- Vereinfacht viele JavaScript Aufgaben
- Führt z.B. einen $-Operator ein

```
window.onload = function(){ alert("welcome");
```



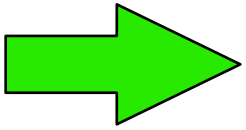
```
$(document).ready(function(){ alert("welcome"); }
```

```
document.getElementById("element").value
```



```
$("element").value
```

```
document.getElementById("element").style.visibiltiy="hidden"
```

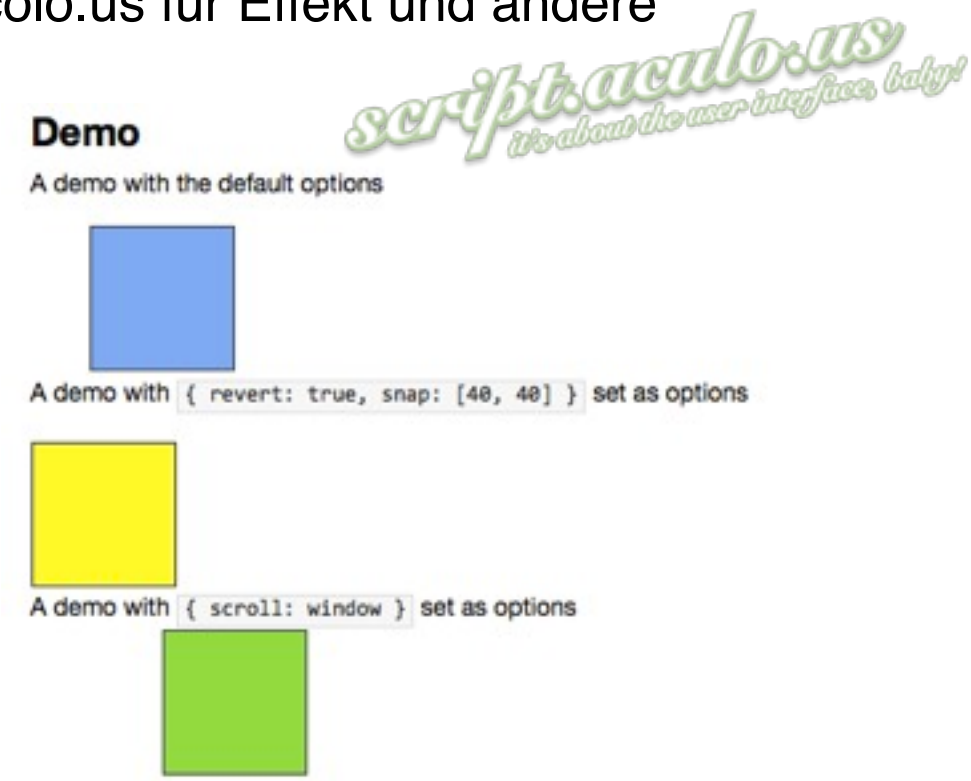


```
$("element").hide(); // auch mit hide("slow");
```

Prototype Zusatzframeworks

- Ganz ähnlich zu jQuery
- Zusätzlich erweiterbar mit script.acolo.us für Effekt und andere Funktionen

- Effect.Appear, Effect.Fade
- Effect.Puff
- Effect.DropOut
- Effect.Shake
- Effect.SwitchOff
- Effect.BlindDown, Effect.BlindUp
- Effect.SlideDown, Effect.SlideUp
- Effect.Pulsate
- Effect.Squish
- Effect.Fold
- Effect.Grow
- Effect.Shrink



http://madrobby.github.com/scriptaculous/combination-effects-demo/ http://madrobby.github.com/scriptaculous/draggable/

So viele Erweiterungen

CSS

DOM

Audio?

Grafik zeichnen?


HTML

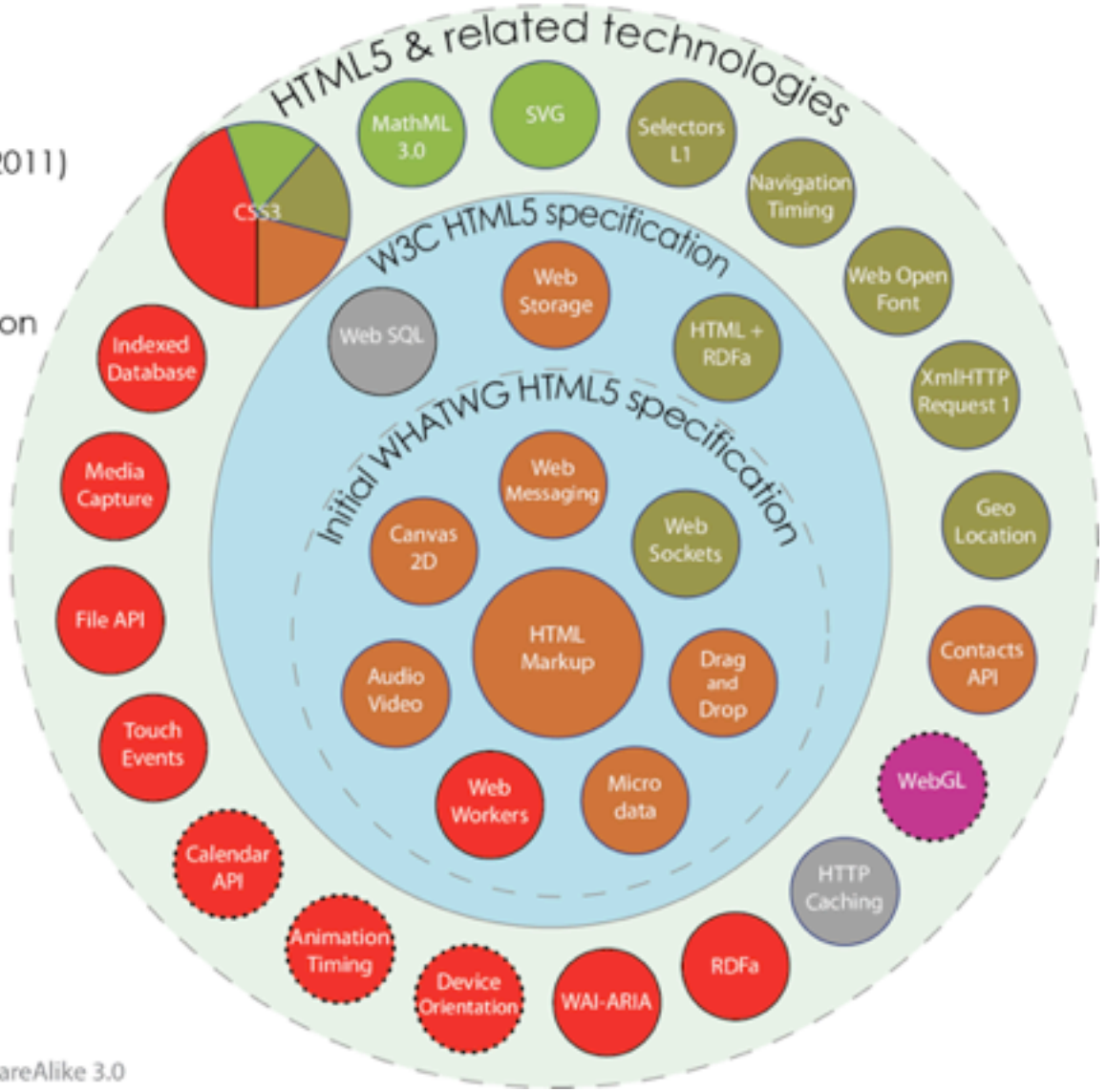
HTML5

- Erweiterung von HTML weg von der Markup-only Sprache
- Mehr Interaktivität, Mehr Möglichkeiten
- Nicht ein großes festgelegtes „Etwas“
- Sondern: Zusammenschluss verschiedener neuer Spezifikationen
- Teilweise noch nicht abgeschlossen
- Browserhersteller fangen mit der Implementierung an
- Wichtige Beispiele:
  - WebSockets
  - Canvas2D
  - Drag&Drop
  - Audio&Video
  - Aber noch viel mehr....

HTML5

Taxonomy & Status (December 2011)

- W3C Recommendation
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated W3C APIs



HTML5 im Test

- Programmierer experimentieren bereits mit HTML5 herum
- Erste Demos werden gesammet
  - Nicht in allen Browsern benutzbar
  - Je nach Implementierungsstand
- www.html5demos.com
- Aber auch schon andere CanvasTests
- http://arcade.rawrbitrary.com/mario/
- http://www.canvasdemos.com/type/games/
- http://www.sinuousgame.com/

10. Interaktive Web-Inhalte

10.1 Clientseitige Web-Skripte: JavaScript

10.2 Dokument-Objekte und DOM

10.3 Serverseitige Web-Skripte

10.4. Beyond JavaScript: Prototype, jQuery, script.aculo.us und HTML5

10.5. Und was ist das? 

Und was ist das?

ecd.html

```
<body>
<div id="replace" style="font-family: Arial; font-size: 30px;
font-weight:bold;"></div>
<script type="text/javascript">
var dt = new Date(2012,01,13,16,00,00);
setTimeout("update()",1000);
function update() {
    var d=new Date();
    var diff = dt.getTime()-d.getTime();
    diff = Math.floor(diff/1000);
    var d = Math.floor(diff/(60*60*24));
    diff-=d*60*60*24;
    var h = Math.floor(diff/(60*60));
    diff-=h*60*60;
    var m = Math.floor(diff/60);
    diff-=m*60;
    var s = Math.floor(diff%60);
    document.getElementById("replace").innerHTML = d+" "+h
+":"+m+":"+s;
    setTimeout("update()",1000);
}
</script>
</body>
```

Viel Erfolg bei der Klausur!

Schöne Ferien!