WebApplications

Programmierung und Design mit HTML5, CSS3 und JavaScript

Universität Regensburg / Fakultät für Physik / Florian Rappl

Vorwort

Dieses Skript wurde für die Vorlesung "Programmierung und Design von WebApplications mit HTML5, CSS3 und JavaScript" geschrieben. Die Bedienung erfolgt über die Tastatur (d.h. die Pfeiltasten, Pos1, Ende, Enter, Leerzeichen, etc. haben eine besondere Bedeutung) oder durch die untere Navigationsleiste. Diese wurde mit JavaScript (mit jQuery) und CSS3 entsprechend verschönert um nicht immer störend im Bild zu sein und die wichtigsten Informationen und Methoden bereitszustellen.

Die Vorlesung setzt nur geringfügige Kenntnisse in HTML voraus. Für CSS sind keinerlei Grundkenntnisse notwendig, allerdings kann vorhandenes Wissen sehr hilfreich sein. JavaScript wird von Grund auf eingeführt - hier sind nur Programmierkenntnisse in einer beliebigen (Skript-) Sprache notwendig, damit die Probleme in der Sprache und nicht in der Programmierung an sich auftreten.

Das Skript wurde während der Vorlesung kontinuierlich erweitert und verbessert. Die Übungsaufgaben sind auf einer separaten Seite, der Seite zum Kurs, welche mit weiteren Materialien zu dieser Vorlesung ausgestattet ist, verlinkt. Die Lösungen zu den Übungsaufgaben sind ebenfalls auf der Kurshomepage zu

finden. Der Übungsbetrieb wird in der zweiten Woche mit grundlegendem HTML beginnen und anschließend mit JavaScript Grundkenntnissen fortfahren.

Neben den HTML5 Fähigkeiten von Desktop-Browsern sind die Möglichkeiten mit den mobilen Browsern im Fokus der Web-Entwickler. Im Internet gibt es jede Menge Webseiten mit Browservergleichen und Demonstrationen von aktuellen Möglichkeiten mit dem neuen Standard. Ein auf jeden Fall aufzurufender Test ist auf der Seite von HTML5Test zu finden. Sollte der verwendete Browser hier zu wenig Punkte holen, wäre ein Browserupgrade für die Vorlesung auf jeden Fall zu empfehlen.

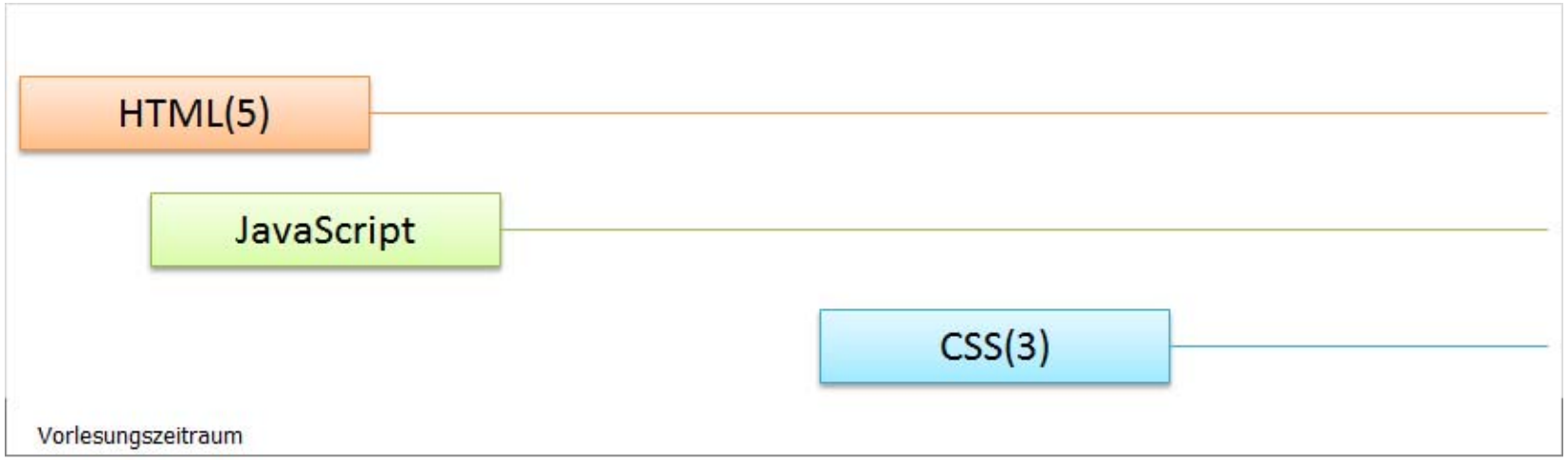
Nachtrag im September 2013 Die Vorlesung hat sich mittlerweile als Blockkurs etabliert. Viele in der Vorlesung vorgestellte Techniken sind mittlerweile sehr weit verbreitet und können bedenkenlos eingesetzt werden. Nachdem der Kurs versucht, an den Grenzen der Technologie zu schreiten, werden daher Technologien wie WebGL stärker berücksichtigt.

Florian Rappl im September 2011 und 2013.

Referenzen:
Die Seite zur Vorlesung [http://html5.florian-rappl.de/]
Vergleich der HTML5 Fähigkeiten von mobilen Browsern [http://mobilehtml5.org/]
Demonstrationen und Browserübersicht zu HTML5 Fähigkeiten [http://html5demos.com/]
Testseite für HTML5-Kompatibilität des eigenen Browsers [http://html5test.com/]

Aufbau der Vorlesung

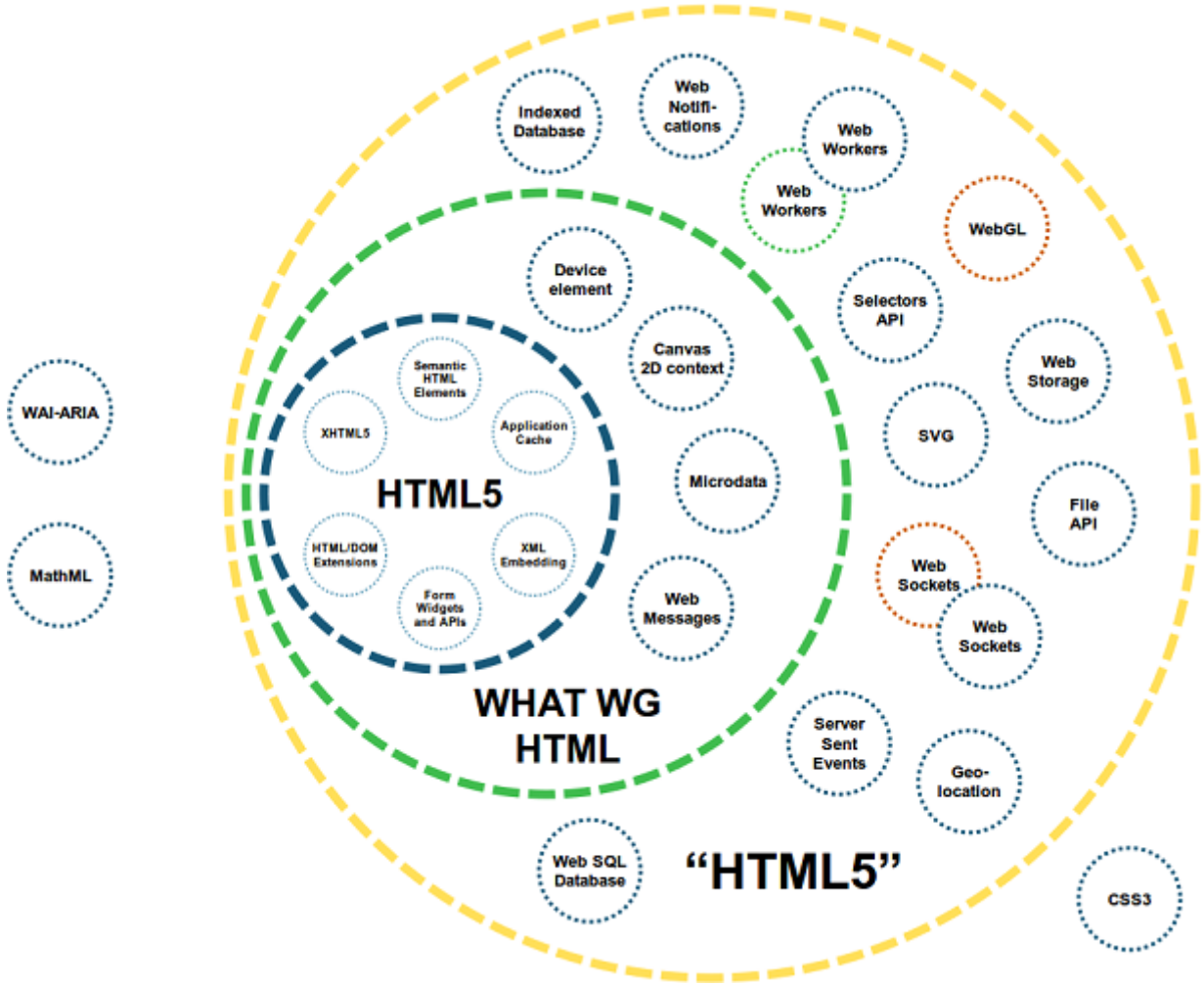
- Vorlesungskriterien so wie auf der Webseite vermerkt
- Übungen im CIP-Pool - ebenfalls wie auf der Webseite vermerkt
- Einstieg über HTML(5), mit viel Übungen in JavaScript
- CSS(3) Anteil wird als letzte Schiene dazu geschaltet



Der grobe Ablauf der Vorlesung

Referenzen:

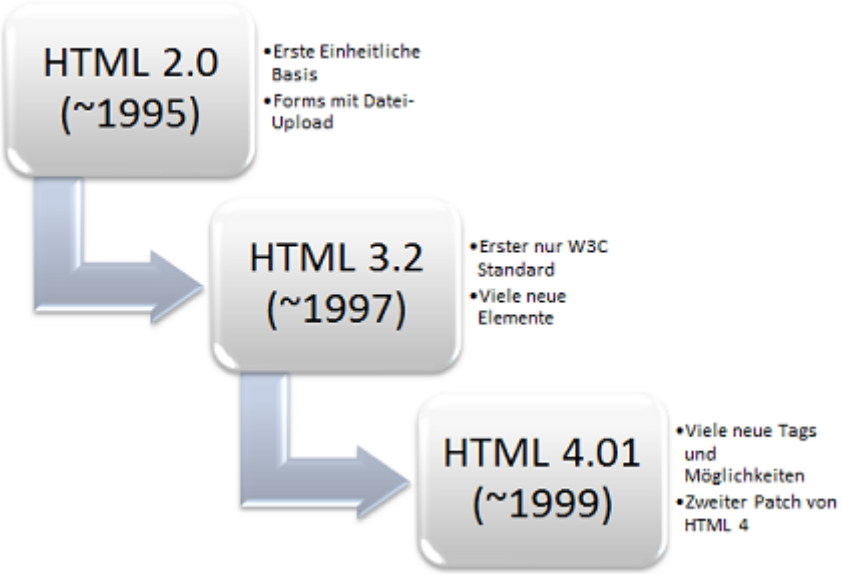
Was ist HTML5?



Einteilung von HTML5 in die verschiedenen Gremien (Grafik von Peter Kröner)

Referenzen:
Die Seite des World Wide Web Consortium (W3C) [http://www.w3.org/]
Die Seite der Web Hypertext Application Technology Working Group [http://www.whatwg.org/]
Webseite von Peter Kröner [http://www.peterkroener.de/]
Codeproject-Artikel zu HTML5 [http://www.codeproject.com/KB/solution-center/HTML5-in-5.aspx]
HTML5 als Alternative zu nativen Anwendungen [http://www.techrepublic.com/blog/programming-and-development/why-html5-makes-justifying-native-applications-more-difficult/4903]

Geschichte von HTML

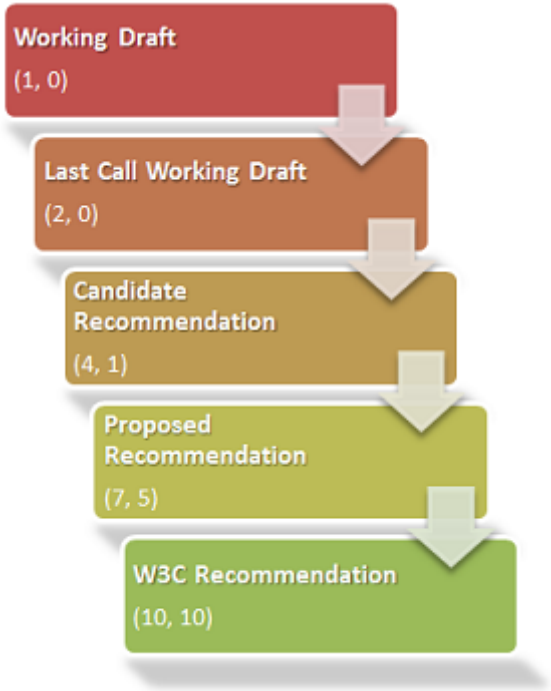


Die grobe Geschichte von HTML

- HTML heißt Hyper-Text Markup-Language und wurde von Tim Berners-Lee 1992 eingeführt
- Die erste standardisierte Version erschien im November 1995 mit HTML 2.0
- Der HTML Standard wird genauso wie CSS, SVG, DOM, SOAP, PNG uvm. vom W3C kontrolliert
- Der erste nur vom W3C eingeführte Standard erschien im Januar 1997 mit HTML 3.2
- Version 3.0 wurde ausgelassen, da dieser bei Einführung bereits veraltet war
- Nach HTML 4.01 dachte man das XHTML (stark an XML angelehnt) die Zukunft sei
- Nach starker Fragmentiertung und viel Kritik wurde dies 2008 eingestellt
- Seit April 2009 wird an HTML5 gearbeitet

Referenzen:
Wikipedia Artikel zu HTML [http://en.wikipedia.org/wiki/Html]
W3C Geschichte von HTML [http://www.w3.org/People/Raggett/book4/ch02.html]
Die frühe Geschichte (1990-1992) [http://infomesh.net/html/history/early/]
Historisches zu HTML [http://www.htmlgoodies.com/tutorials/html_401/html4-ref/article.php/3460261/A-Brief-History-of-HTML.htm]
Geschichte des Internets (HTML) [http://inventors.about.com/od/computersoftware/a/html.htm]
Sehr guter Artikel auf Tutsplus [http://net.tutsplus.com/articles/general/a-brief-history-of-html5/]

Arbeitsweise des W3C



Der Standardisierungsprozess des W3C - von Rot (nicht verwenden) bis Grün (verwendbar) sind die einzelnen Schritte aufgezeigt. In Klammer steht (Browsersupport, Einsetzbar auf prof. Webseiten) auf einer Skala von 0 (sehr schlecht) bis 10 (sehr gut).

- Zunächst kann jeder einen neuen Vorschlag einbringen
- An diesem kann dann gearbeitet werden bzw. er wird diskutiert
- Irgendwann gibts dann einen Last Call (letzte Möglichkeit für Diskussionen)
- Sollte der Antrag immer noch OK sein, so wird dieser eine Candidate Recommendation (CR)
- Fast alles was wir betrachten ist eine Candidate Recommendation oder höher
- Bei Features unter einer Proposed Recommendation (PR) sollte man vorsichtig sein
- W3C Recommendation wird etwas erst nach Jahren
- Entscheidend hierfür ist die Verbreitung und die Unterstützung des Elements

Referenzen:
W3Schools Tutorial zum W3C [http://www.w3schools.com/w3c/default.asp]
Wikipedia Artikel zum W3C [http://en.wikipedia.org/wiki/World_Wide_Web_Consortium]
Die Seite des W3C [http://www.w3.org/]
Die deutsche Seite des Konsortium [http://www.w3c.de/]

Um was gehts hier genau?

Wir werden uns mit drei Technologien beschäftigen:

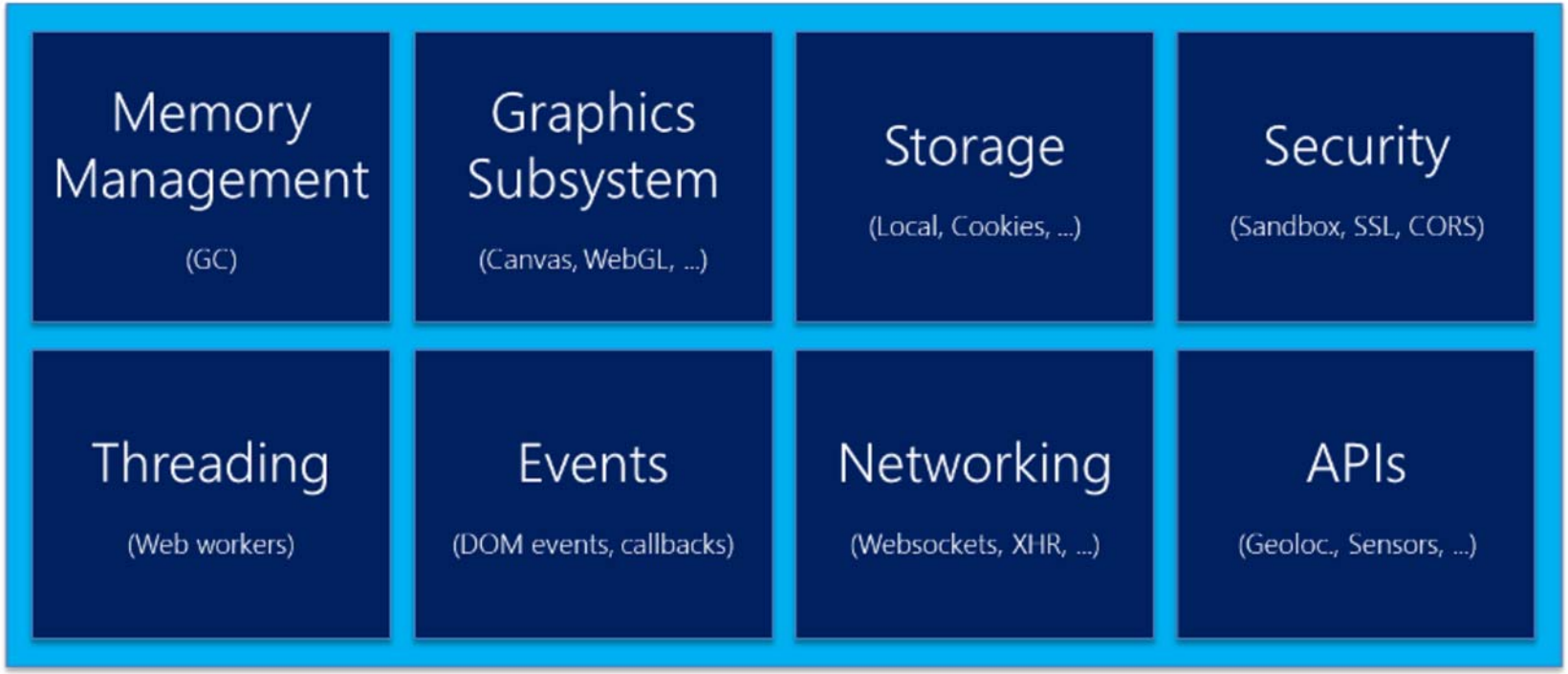
- HTML(5) zur Beschreibung der Seite (Design)
- JavaScript für die eigentlichen Interaktionen (Programmierung)
- CSS(3) zur Gestaltung dieser (Design und Programmierung)

Wie bereits erwähnt werden diese Technologien schrittweise eingeführt. Daneben werden wir uns bei Zeit noch mit folgenden Möglichkeiten beschäftigen:

- Einsatz von jQuery zur Vereinfachung des DOM (Document Object Model) Zugriffs mit JavaScript
- Bereits fertige UI Fähigkeiten sehr leicht mit jQuery UI einbinden
- Spiele und Animationen / Grafiken (auch 3D) mit JavaScript
- Abwärtskompatibilität zu älteren Webbrowsern schaffen
- Webseiten durch einfache Tricks optimieren (weniger Requests, JavaScript minifizieren, ...)
- JavaScript für die Client- und Serverentwicklung mit **node.js**

Referenzen:
HTML5 Tag Referenz [http://www.w3schools.com/html5/html5_reference.asp]
CSS3 Kompendium mit jeder Menge Tricks [http://www.css3.com/]
JavaScript / DOM Referenz [http://www.w3schools.com/jsref/default.asp]
node.js [http://nodejs.org/]

Warum Webapplications?



Der Browser ist ein eigenständiges System

- Ein Browser erfüllt alle Anforderungen an ein Betriebssystem
- Webanwendungen zu schreiben gibt uns viele Vorteile, die wir nützen möchten

Referenzen:
Wikipedia Virtuelle Maschine [http://de.wikipedia.org/wiki/Virtuelle_Maschine]
Wikipedia Betriebssystem [http://de.wikipedia.org/wiki/Betriebssystem#Aufgaben]

Der Nutzen von HTML5

- HTML5 bedeutet: Die Wiedervereinigung von HTML mit neuen (längst überfälligen) Elementen (Tags)
- HTML5 bedeutet nicht: Eine völlig neue Art der Erstellung von Webseiten
- Für uns bringt HTML5 jede Menge neuer nützlicher Elemente mit sich wie z.B. neue Forms Elemente, die sich auch selbstständig validieren können und somit jede Menge JavaScripts sparen (können)
- Allgemein bringt HTML5 genauere Regeln zur Implementierung in den Browsern mit sich - dadurch werden in Zukunft die syntaktischen Unterschiede zwischen den Browsern deutlich verringert
- Das bedeutet, dass das Parsing Modell ist nun standardisiert und Fehler werden von allen Browsern gleich interpretiert werden
- Durch `audio`, `video` und das `canvas`-Element sind nun reichhaltige Multimedia-Fähigkeiten ohne zusätzliche Plug-Ins möglich

- Durch neue Tags wie `header`, `footer`, `section`, `article`, nav, uvm. können Screenreader die Seite effektiver auslesen und anzeigen
- HTML5 bietet sehr viele neue JavaScript Objekte, welche WebApplications erst möglich machen

Referenzen:
Die Seite des World Wide Web Consortium (W3C) [http://www.w3.org/]
Die Seite der Web Hypertext Application Technology Working Group [http://www.whatwg.org/]
Die Seite der W3C HTML5 Arbeitsgruppe [http://www.w3.org/TR/html5/]

Besseres Cross-Browsing

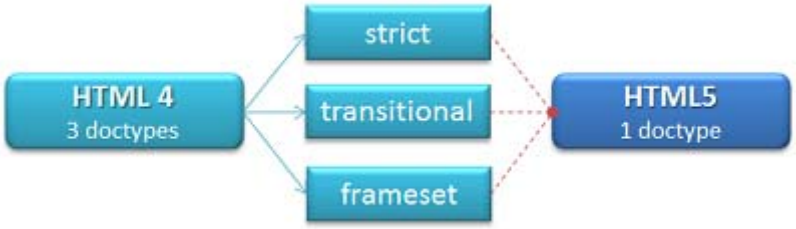
Die entscheidenden Mitglieder der WHATWG mit der Unterstützung der Features in der aktuellen Version (Stand: September 2011)

- Browser sind unser Portal in alle Plattformen (Windows, Linux, ...) und Systeme (PCs, Mobiltelefone, ...)
- Ein wichtiges Ziel von HTML5 ist: Was auf einem Browser funktioniert, sollte auf allen klappen
- Der Standard wird von den obigen fünf Herstellern festgelegt - ohne Einigung kein Standard[1]
- Der einzige europäische Teilnehmer ist der Pionierbrowser von Opera (Rest: USA)
- Früher ein Problem: Was passiert bei einem Fehler? Was ist ein Fehler? Mit HTML5 ist dies standardisiert
- Sehr wichtig für uns werden die Debugging-Fähigkeiten der Browser werden - hier gibt es Unterschiede

Referenzen:
Der Microsoft Internet Explorer [http://windows.microsoft.com/en-US/internet-explorer/products/ie/home]
Google Chrome [http://www.google.com/chrome]
Mozilla Firefox [http://www.mozilla.org/en-US/firefox/new/]
Apple Safari [http://www.apple.com/safari/]
Opera [http://www.opera.com/]

[1] So konnte leider keine Einigung bei einem Videoformat erzielt werden - daher gibt es dort keinen Standard

Ein Standard, ein Doctype



HTML 4 wollte durch drei Doctypes eine Abwärtsfähigkeit schaffen - HTML5 schafft dies mit nur einem Doctype

- HTML 4 hat drei verschiedene Doctypes:
  - `strict`, was CSS am genausten unterstützt und einen XML ähnlich strikten Syntax fordert
  - `transitional` um auch ältere Elemente, sowie einen eher laschen Syntax zu unterstützen
  - `frameset` zur Unterstützung von `<frame>` Elementen (ansonsten analog zum vorherigen)
- Dies sorgte neben dem (unnötig) langen Doctype zu Verwirrung und Unmut
- Uneinigkeit herrschte ebenfalls: Opera fand rund 15000 verschiedene Doctypes auf 2 Mio. Webseiten
- Ein möglicher Doctype von HTML 4:

```
01. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
```

Referenzen:
W3Schools über HTML Doctypes [http://www.w3schools.com/html/html_doctype.asp]
W3Schools über HTML5 Doctype [http://www.w3schools.com/html5/tag_doctype.asp]
Wahl eines Doctypes (HTML 4) [http://htmlhelp.com/tools/validator/doctype.html]
Blog über den HTML5 Doctype [http://themaingate.net/dev/html/all-you-need-is-doctype-html]

Änderungen zu HTML 4

Im Vergleich zu HTML 4 wurden v.a. neue Elemente und Attribute eingeführt, sowie HTML genauer spezifiziert. Dies bedeutet:

- Es gibt nun nicht nur standardisierte Tags, sondern auch auch ein festgelegtes Verhalten
- Statt mehrere unterschiedliche Doctypes zu haben, die unterschiedliche Browsermodi aufrufen (oder nicht), gibt es nun ebenfalls einen standardisierten (HTML5) Doctype
- Viele bekannte Elemente sind einfacher zu schreiben, da ein Standard festgelegt wurde - so sind Skriptelemente standardmäßig auf die Sprache JavaScript festgelegt und Stylesheet Elemente auf CSS

Wie sieht das ganze nun in der Praxis aus?

```
01. <!DOCTYPE html> <!--Der neue DOCTYPE - endlich mal einer, den man sich merken kann-->
```

```
<html lang="de"> <!--HTML ist immer noch unser Obertag - Möglichkeit die Sprache des Dokuments festzulegen-->
<head>
<meta charset="utf-8" /> <!--Direkt schließen à la XML ist OK - muss aber nicht sein-->
<script src="test.js" > </script> <!--Schließen à la leerer HTML Tag geht auch - hier muss-->
</head>
<body> <!--Ohne einen Body Tag funkioniert z.B. im IE kein CSS Styling-->
...
</bodY> <!--Die Schreibweise der Attribute und Tags ist nicht case-sensitive-->
</html>
```

Referenzen:
Änderung aus Sicht des W3C [http://www.w3.org/TR/html5-diff/]
Seite mit Infos zum Konvertieren von HTML 4 Seiten nach HTML5 [http://www.html-5.com/tutorials/converting-to-html-5.html]
Zusammenfassung der Änderungen bei Stackoverflow [http://stackoverflow.com/questions/134727/whats-the-key-difference-between-html-4-and-html-5]
Zehn essentielle Unterschiede bei Developerdrive [http://www.developerdrive.com/2011/08/10-essential-differences-between-html4-and-html5/]

Entfernt und neu

Offiziell wurde entfernt:

- <font> (CSS benutzen!)
- <applet>, statt dessen <embed> benutzen
- <big> (CSS ...)
- <blink> (ebenfalls CSS)
- (ebenfalls styling)
- <marquee> (Animationen über JS, CSS)
- <frame>, aber <iframe> bleibt

Neue Elemente (u.a.):

- (Hervorhebung)
- <time> (Zeit)
- <blockquote> und <cite>
- <details> (Detailliste)
- <fieldset> (Formular)

- `<figure>` und `<figcaption>`
- `<dl>` (Definition)
- `<mark>` (Markierung)
- `<ruby>` mit `<rp>` und `<rt>`

... sowie weitere (~90). Diese sind aber entweder sehr speziell oder noch nirgends unterstützt.
Referenzen:
W3Schools - Alle HTML5 Tags [http://www.w3schools.com/html5/html5_reference.asp]
IBM Informationen zu den neuen Tags [http://www.ibm.com/developerworks/library/x-html5/]
In HTML5 entfernte Tags [http://www.mywindowsclub.com/resources/4736-HTML-tag-reference-Tags-removed-or-obsolete.aspx]
Entfernte Attribute und Tags in HTML5 [http://www.tutorialspoint.com/html5/html5_deprecated_tags.htm]

Neue Grundelemente

- HTML5 führt einige neue Elemente ein, welche direkt nichts anderes als ein `div` darstellen

Schemenhafter Aufbau einer Webseite mit den neuen Elementen

- Der Vorteile in der Benutzung dieser neuen Tags liegen auf der Hand:
  - Bessere Auslesbarkeit durch Screenreader
  - Strukturierterer Aufbau
  - Direkteres und damit auch leichteres Styling
- Die Abwärtskompabilität ist bis auf den IE sehr gut gegeben[1]

Referenzen:
Der HTML Header Tag im Detail (für Titel) [http://www.w3schools.com/html5/tag_header.asp]
Der HTML Footer Tag im Detail (für Fußzeilen) [http://www.w3schools.com/html5/tag_footer.asp]
Der HTML Section Tag im Detail (für Sektionen) [http://www.w3schools.com/html5/tag_section.asp]
Der HTML Article Tag im Detail (für Artikel) [http://www.w3schools.com/html5/tag_article.asp]
Der HTML Nav Tag im Detail (für Navigation) [http://www.w3schools.com/html5/tag_nav.asp]

Der HTML Aside Tag im Detail (für Randnotizen) [http://www.w3schools.com/html5/tag_aside.asp]

[1] Für ältere IEs gibt es jedoch über JavaScript einen sehr leichten Trick (später)

Semantischer Aufbau

- HTML 4 repräsentierte den Stand der Webentwickler um 1997, HTML5 repräsentiert den Stand von heute
- Insgesamt 28 neue Elemente (viele davon sind offensichtlich von ID und Klassenamen inspiriert worden)
- Ein Ziel von HTML5 ist es, Elementen mehr (direkt ersichtliche) Bedeutung zu geben
- Browser können nun die Informationen in einer Webseite besser interpretieren

Die Wichtigkeit des semantischen Aufbaus von HTML5 sieht man bereits am Logo-Badge (CSS3 links, Semantics rechts)

- Alle Elemente können natürlich öfters verwendet werden (mehrere `nav`, `header`, `hgroup`, `footer`, `aside`, `article`, `section`, ... Elemente möglich)
- Die Aussage bzw. Struktur der Elemente sollte passen - dies erleichtert die Zusammenarbeit Webbrowser - Webseite
- Für uns von großer Bedeutung: Crawler / Screenreader können mehr über die Seite erfahren

Referenzen:
Erklärung des semantischen Aufbaus [http://www.alistapart.com/articles/semanticsinhtml5]
Hilfe zur korrekten Anwendung der semantischen Elemente [http://www.codeproject.com/KB/HTML/semanticHtml5PageLayout.aspx]
Historisches zur Entwicklung der semantischen Beschreibung [http://www.html5tutorial.info/html5-semantic.php]
Smashing Magazin über semantische Tags [http://coding.smashingmagazine.com/2011/11/18/html5-semantics/]
Ein weiterer Artikel über die Wichtigkeit von semantischen Tags [http://coding.smashingmagazine.com/2011/11/12/pursuing-semantic-value/]
Kritischer Artikel über semantische Tags [http://coding.smashingmagazine.com/2011/11/11/our-pointless-pursuit-of-semantic-value/]

Wann genau verwenden?

- Für das `header` Element gilt:
  - Sollte das erste Element mit Inhalt auf der Seite sein
  - Enthält im Regelfall Logos, Titel, Such- und Loginformulare sowie essentielle Links (z.B. Homepage)
  - Man sollte `hgroup` einsetzen um zusätzliches Markup vom Titel abzugrenzen
- Die Spezifikation für den `nav` Tag sagt:
  - Markiert eine Navigationsgruppe
  - Das `nav` Element sollte nicht für jede Sammlung von Links, z.B. Sponsorenlinks, verwendet werden
  - Eine Suchergebnisseite stellt auch kein `nav` dar, da hier die Suchergebnisse den Inhalt darstellen

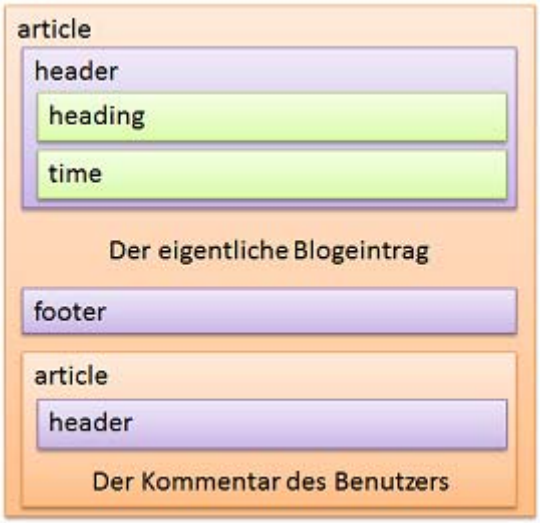
- Der `footer` dient für folgende Zwecke:
  - Letztes Element mit Inhalt (z.B. weiterführende rechtliche Links und Hinweise)
  - Hier können Informationen über den Autor (`address`), das Erstellungsdatum (`time`) sowie Links platziert werden

Referenzen:
HTML5 Tutorials zu den semantischen Tags [http://www.html5tuts.co.uk/tutorials/semantics/]
W3C Dokument zu den semantischen Tags [http://dev.w3.org/html5/spec/Overview.html]
Adobe zum Thema semantische Elemente [http://www.adobe.com/devnet/dreamweaver/articles/understanding-html5-semantics-pt2.html]

Konkretes Beispiel

Das Grundgerüst für einen **Blogeintrag**:

```
01. <article> <!-- Blogeintrag -->
02. <header> <!-- Titelzeile -->
03. <h2>Physics of the future</h2>
04. <time datetime=2011-10-13 pubdate>13. Oktober 2011</time>
05. </header>
06. <p>Michio Kaku's Buch ist jetzt schon ein Klassiker ohne Ablaufdatum.</p> <!--Text-->
07. <footer> <!-- Abschließendes -->
08. Veröffentlicht in <a href="/?cat=3">Bücher</a> von <address>Anonymous</address> mit <a href="/?p=34#respond">1 Kommentar</a>
09. </footer>
10. <article> <!-- Kommentar -->
11. <header>
12. Kommentar von <a href="http://www.florian-rappl.de">Florian Rappl</a> um <time datetime="2011-10-14T08:45Z">8:45 Uhr am
    14.10.2011</time>
13. </header>
14. <p>Hört sich gut an - das muss ich mir mal durchlesen!</p>
15. </article> <!-- Ende des Kommentares -->
16. </article> <!-- Ende des Blogeintrags -->
```



Schema unseres Codes

Einstieg in JavaScript

- JavaScript hat gar nichts mit Java gemein - der ursprüngliche Name ist LiveScript
- Ziel und Zweck: Fähigkeiten von Webseiten verbessern und eine dynamische Webseite auf dem Client zu erzeugen
- JavaScript kann in einem HTML Dokument geschrieben oder von extern eingebunden werden
- Ein Beispiel für direkte Einbindung (im Dokument):

```
01. <!--In HTML5 ist JavaScript Standard, daher brauchen wir KEINE weiteren Angaben wie language-->
02. <script>/*JavaScript Kommentar*/</script>
```

- Ein Beispiel für indirekte (externe) Einbindung:

```
01. <script src="path/to/beispiel.js"></script>
```

- Durch den Wettkampf unter den Browser Herstellern werden die JavaScript-Engines immer schneller
- Am Anfang werden wir Ausgaben in unserem Code immer über die eingebaute Methode `alert()` ausgeben

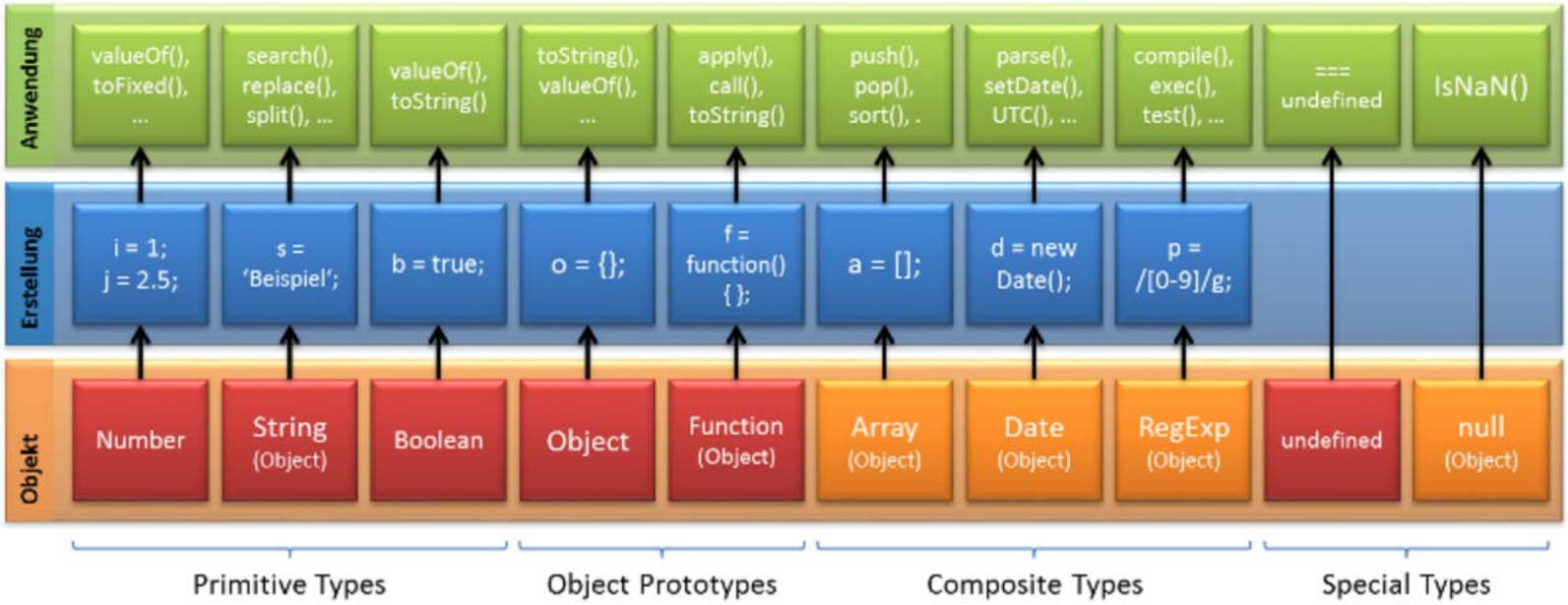
Referenzen:
Wikipedia Artikel über JavaScript Engines [http://en.wikipedia.org/wiki/JavaScript_engine]
W3Schools Referenzen und Tutorials zu JS [http://www.w3schools.com/js/]
JavaScript Anfängerhilfe [http://www.irt.org/articles/js083/]

JavaScript verstehen

- Objekte
  - Objekte kapseln Eigenschaften und Methoden (Schnittstelle!)
  - Objekte ermöglichen uns Zugriff auf Elemente einer Webseite
  - z.B. können wir mit dem `window` Objekt auf den aktuellen Kontext zugreifen
- Methoden
  - Aktive Schnittstelle eines Objektes (zum Ausführen einer Aktion)
  - Wenn unabhängig vom Kontext kann man von einer Funktion sprechen
  - z.B. schließt `window.close()` das aktuelle Fenster (Tab) komplett
- Eigenschaften
  - Verkörpern Merkmale des Objektes (sog. passive Schnittstelle)
  - Eigenschaften geben Informationen über den Zustand des Objektes
  - z.B. setzt `document.bgColor = "FF0000"` die Hintergrundfarbe auf rot

Referenzen:
Wikipedia Artikel über JavaScript [http://de.wikipedia.org/wiki/JavaScript]
Informationen über das window Objekt [http://www.w3schools.com/jsref/obj_window.asp]
Informationen über das document Objekt [http://www.w3schools.com/jsref/dom_obj_document.asp]
OOP in JavaScript erklärt [http://mckoss.com/jscript/object.htm]

Variablen und Datentypen



Auflistung der verschiedenen JavaScript Datentypen

- Es gibt 6 elementare Typen: `Number`, `String`, `Boolean`, `undefined`, `Function` und `Object`
- Alle anderen Typen (z.B. `Array`) bauen auf `Object` auf (auch `null`)
- Unter Variablen verstehen wir Speicher für Objekte und Werte (z.B. gilt Zahl = Wert und String = Objekt)
- Deklaration einer Variablen erfolgt mit ihrer ersten Aufzählung (Lokal: `var` davor, z.B. `var i = 2`)

Referenzen:
O'Reilly JavaScript Tutorial (Datentypen) [http://oreilly.com/javascript/excerpts/learning-javascript/javascript-datatypes-variables.html]
Einführung in JavaScript / Datentypen [http://www.sislands.com/coin70/week1/datatype.htm]
MSDN Hilfe über JavaScript Datentypen [http://msdn.microsoft.com/en-us/library/7wkd9z69(v=vs.94).aspx]
Crockford on JavaScript - Volume 1: The Early Years [http://www.youtube.com/watch?v=JxAXlJEmNMg]

Arrays (1)

- Damit sich hinter einer Variablen mehr Objekte verstecken können, muss es sich um ein Array handeln
- Arrays können auf sehr unterschiedliche Art und Weise initialisiert werden:

```
01. var leer1 = []; //Empfohlen
02. var leer2 = new Array(); //Nicht empfohlen
03. var reserviert = new Array(14); //Bringt keine Performance-Vorteile
```

```
04. var belegt1 = new Array('Milch', 'Zucker', 'Mehl');
05. var belegt2 = ['Milch', 'Mehl', 'Äpfel'];
```

- Möglichkeiten zum Hinzufügen eines Elements mit `push()` (ans Ende) und `unshift()` (an den Anfang)
- Der Zugriff auf Array Elemente erfolgt über eckige Klammern, z.B.

```
01. var lang = ['C#', 'Java', 'C++'];
02. var java = lang[1];
03. var test = lang[3];//undefined
```

- Der Zugriff ist 0 basiert, wie in anderen C ähnlichen Sprachen
- Bei nicht-definierten Zugriffen gibt es keinen Fehler, sondern `undefined` als Rückgabe

Referenzen:
W3Schools über das Array Objekt [http://www.w3schools.com/jsref/jsref_obj_array.asp]
Profiwissen über JS Arrays [http://www.hunlock.com/blogs/Mastering_Javascript_Arrays]

Arrays (2)

- Die Eigenschaft `length` gibt eine Zahl zurück, welche 1 größer als der größte Integer Key ist
- Mit `sort()` können wir das Array sortieren – optional unter Angabe einer Sortierfunktion
- Standardmäßig wird nur die Stringdarstellung sortiert, z.B.

```
01. var a = [1, 5, 12, 23, 100];
02. a.sort();//1 100 12 23 5
03. a.sort(function(a,b) {
04.     return a-b;
05. }); //1 5 12 23 100
```

- Zum Entfernen mit `splice()` wird der Index und die Anzahl benötigt oder wir löschen das letzte mit `pop()`

```
01. var kundenliste = []; //Erstellt ein leeres neues Array mit length=0
02. kundenliste.push('Herbert Maier'); //Ein neuer Eintrag drin
03. kundenliste.push('Max Hilbert', 'Anton Mustermann'); //Zwei weitere hinzugefügt
04. var i = kundenliste.length; //Variable i hat nun den Wert 3
05. kundenliste.splice(0, 2); //Entfernt zwei Einträge ab Index 0 - es bleibt 'Anton Mustermann'
```

```
06. var j = kundenliste.length; //Variable j hat nun den Wert 1
```

Referenzen:
MDN über Array.sort() [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort]
Mehr über splice() [http://www.w3schools.com/jsref/jsref_splice.asp]
Profiwissen über JS Arrays [http://www.hunlock.com/blogs/Mastering_Javascript_Arrays]

Operatoren

- Operatoren sind reservierte Zeichen, die Operationen (z.B. Addition) ausführen
- Es gibt: Logik-, Arithmetik-, Bit- und Vergleichsoperatoren (sowie Condition ? und Assignment =)
- Genau wie in anderen Sprachen gibt es bei manchen Operatoren Prä- (`++i`) und Suffixdarstellungen (`i++`)
- Es gibt zu beachten - JavaScript nimmt einiges nicht so genau, z.B. ist `'0'` gleich `0` bei `==`
- Daher gibt es zwei weitere Operatoren (`===` und `!==`) - diese vergleichen Wert **und** Typ
- Außerdem ist die `0` nicht vorzeichenlos - dies kann zu verwirrenden Ergebnissen führen (`a/0 != a/(-0)`)

Arithmetische Operatoren

Op.	Bedeutung	Beispiel
`+`	Addition	`3 + 4` ergibt `7`
`++`	Inkrement	`i=3;i++` ergibt `4`
`-`	Subtraktion	`3 - 4` ergibt `-1`
`--`	Dekrement	`i=3;i--` ergibt `2`
`*`	Multiplikation	`3 * 4` ergibt `12`
`/`	Division	`3 / 4` ergibt `0.75`
`%`	Modulo	`5 % 4` ergibt `1`

Vergleichsoperatoren

Op.	Bedeutung	Beispiel
`==`	gleich	`3 == '3'` ergibt `true`
`!=`	ungleich	`3 != 3` ergibt `false`
`===`	typgleich	`3 === 3` ergibt `true`
`!==`	typungleich	`3 !== '3'` ergibt `true`
`<`	kleiner	`3 < 4` ergibt `true`
`<=`	kleiner gleich	`4 <= 3` ergibt `false`
`>`	größer	`3 > 4` ergibt `false`
`>=`	größer gleich	`4 >= 3` ergibt `true`

Weitere Operatoren

Op.	Bed.	Beispiel
`&&`	und	`true && false` ergibt `false`
`||`	oder	`true || false` ergibt `true`
`!`	nicht	`!false` ergibt `true`
`&`	bit und	`2 & 1` ergibt `0`
`|`	bit oder	`2 | 1` ergibt `3`
`~`	bit nicht	`~2` ergibt `-3`

Referenzen:
Operatoren bei W3Schools [http://www.w3schools.com/js/js_operators.asp]
WikiBooks über JS-Operatoren [http://en.wikibooks.org/wiki/JavaScript/Operators]
JavaScript-Operator Profiwissen [http://www.timmywillison.com/pres/operators/#landing]

Konvertieren mit Operatoren

- Um eine boolsche Variable zu erhalten, ist doppelte Negation, d.h. `!!myvar` ausgezeichnet
- Strings kann man allgemein durch Addition mit einem String, z.B. `myvar + ''`, erhalten
- Alle Typen konvertieren zu Zahlen bei (fast) allen Operationen, z.B. `myvar * 1` oder `+myvar`

- Wir können solche Zahlen auch runden, hier ist `~~myvar` geeignet
- Neben boolschen Werten, Zahlen und Strings können wir auch Standardwerte definieren
- So gibt `myvar || mydefault` immer `mydefault` zurück, außer `myvar` konvertiert zu `true`
- Umgekehrt gibt `myvar && mydefault` immer `myvar` zurück, außer `myvar` konvertiert zu `true`
- Sinnvoll vor Konvertierungen sind auch die Operatoren `typeof` und `instanceof`, z.B.

```
var o = {};//Instanz eines Objekts erstellt (prototype: Objekt)
var f = function() {};//Instanz einer Funktion erstellt (prototype: Objekt)
o instanceof Object;//true
typeof o;//"object"
f instanceof Object;//true
typeof f;//"function"
```

Referenzen:
JavaScript Type Konvertierungen [http://jibbering.com/faq/notes/type-conversion/]
MDN zu typeof [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/typeof]
MDN zu instanceof [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/instanceof]

Beispiel (1)

```
//Wir arbeiten mit einem Array (arr)
var arr = [];
arr.push(1);
arr.push('2');
arr.push(true);
arr.push(1.5);
arr.push(2.5);
arr.splice(1,2);
//Ausgabe der Eigenschaft Länge
alert('Die Länge des Arrays ist: ' + arr.length);
//Operationen - Ausgabe über Alert im aktuellen Fenster
alert('Addiere die beiden Elemente zusammen: ' + (arr[0] + arr[1]));
alert('Hier vorsicht: ' + arr[0] + arr[1]);
//Ein Fenster aufmachen und manipulieren
```

```
15. var win = window.open();
16. win.document.write('Das ist ein Beispiel!');
17. win.document.bgColor = "FF0000";
18. //Ausgabe von alert im neuen Fenster - modaler Dialog!
19. win.alert('Test');
20. win.close();
```

Kontrollstrukturen

- Bis jetzt geht unser Code immer linear runter – keine Verzweigung möglich
- Viele (z.B. aus C) bekannte Schlüsselwörter gibt es auch hier:
  - `for(_;_;_) {...}` für eine Zählerschleife (Vor Ausführung, Bedingung, Ende d. Iteration)
  - `for(...) {...}` für eine implizite Zählerschleife ("foreach"), z.B. `for(var name in names)`
  - `while(_) {...}` für eine kopfgesteuerte Schleife (Bedingung)
  - `do {...} while(_);` für eine fußgesteuerte Schleife (Bedingung)
  - `switch(_) { case _: ... break; }` für eine Fallunterscheidung (Variable, Wert)
  - `if(_) {...} else {...}` für eine bedingte Anweisung (Bedingung)
- Die Bedingung wird als boolscher Wert abgefragt – also `true` oder `false`
- Bedingte Anweisungen können analog zu allen C ähnlichen Sprachen beliebig verschachtelt werden
- Um in Fallunterscheidungen einen Standardblock zu definieren, verwendet man anstelle von `case` das Schlüsselwort `default`

Referenzen:
Erklärungen zum JavaScript foreach [http://pietschsoft.com/post/2008/02/28/JavaScript-ForEach-Equivalent.aspx]
If/Else in JavaScript [http://www.w3schools.com/js/js_if_else.asp]
Fallunterscheidungen mit switch [http://www.javascriptkit.com/javatutors/switch.shtml]
Details über Schleifen [http://james.padolsey.com/javascript/looping-in-javascript/]

Funktionen

- Diese werden über das Schlüsselwort `function` erstellt, z.B. `function foo() {...}`
- Besteht aus abgeschlossenem Block mit Anweisungen und (möglichen) Parametern
- Ziel ist die Kapslung von Anweisungen zur (variablen) Ausführung an beliebigen Stellen
- Jeder Parameter ist in JavaScript optional – daher können Funktionen nicht überladen werden
- In einer Funktion kann man über `arguments` (Array) auf alle übergebenen Parameter zugreifen

- Den Parametern Standardwerte zuzuweisen ist unsere Aufgabe – unbedingt beachten
- Funktionsaufrufe geschehen über den Funktionsnamen und den Parameterwerten in Klammern
- Es ist möglich Methoden ohne anonyme Funktion zu erstellen (können direkt ausgeführt werden)
- JavaScript kennt keine explizit genannten Rückgabewerte, da die Sprache sowieso dynamisch ist
- Standardmäßig wird `undefined` zurückgegeben – außer man gibt mit `return` etwas zurück
- Wird in einer Funktion auf einen (bzw. den aktuellen) Kontext verwiesen, so ist dies eine Methode

Referenzen:
SelfHTML über Funktionen in JS [http://de.selfhtml.org/javascript/objekte/function.htm]
Tutorial zu Funktionen [http://www.tizag.com/javascriptT/javascriptfunction.php]
Anonyme Funktionen (Wikibooks) [http://en.wikibooks.org/wiki/JavaScript/Anonymous_Functions]
Optionale Parameter [http://www.tipstrs.com/tip/354/Using-optional-parameters-in-Javascript-functions]
Fünf Wege eine Methode aufzurufen [http://devlicio.us/blogs/sergio_pereira/archive/2009/02/09/javascript-5-ways-to-call-a-function.aspx]

Beispiel (2)

```
01. function example2() {
02.     var p = prompt('Welche Funktion aufrufen? (fibonacci, fib, sqrt, cos, sin)', 'fibonacci');
03.     switch(p) {
04.         case 'sin':
05.             var k = Math.sin(prompt('Argument für Sinusfunktion:', 144));
06.             break;
07.         //...
08.         default:
09.             var k = fibonacci(prompt('Wert für Fibonacci Funktion angeben:', 0));
10.     }
11.     //...
12. }
13. function fibonacci(n) {
14.     //...
15.     var t = 1, tp = 0;
16.     for(var i = 2; i <= n; i++) {
17.         var tmp = t; t = fib(t, tp); tp = tmp;
18.     }
19.     return t;
```

```
20. }
21. //...
```

JavaScript in HTML nutzen

- Einbindung erfolgt wie erwähnt über das `<script>` Element
- Allgemein wird JavaScript immer über sog. Ereignisse angesprochen
- Je nachdem um welches HTML Element es sich handelt, stehen andere Ereignisse zur Verfügung
- Beispiel: Bei vielen Elementen existiert das `onclick` Event, das beim Klicken ausgeführt wird:

```
01. <button onclick="alert('Das ist ein Test!');">Teste mich!</button>
```

- Daneben können wir mit JavaScript noch Werte aus HTML auslesen und manipulieren:
  - Die Seite ist im `document` Objekt gekapselt (hier ist z.B. auch das <body> Element als Objekt enthalten)
  - Um im `document` auf benannte Elemente (z.B. `<form name="eingabe" ... />`) zuzugreifen schreiben wir `document.getElementsByName('eingabe')[0]` (hierbei ist `[0]` ein Array-Zugriff auf das 1. Element)
  - Der Array-Zugriff ist notwendig, da wir mehrere Elemente zurück erhalten (können)
  - Um auf Elemente mit einer ID (z.B. `<div id="content" ... >`) zuzugreifen benutzt man `document.getElementById('content')` (die ID muss eindeutig sein)

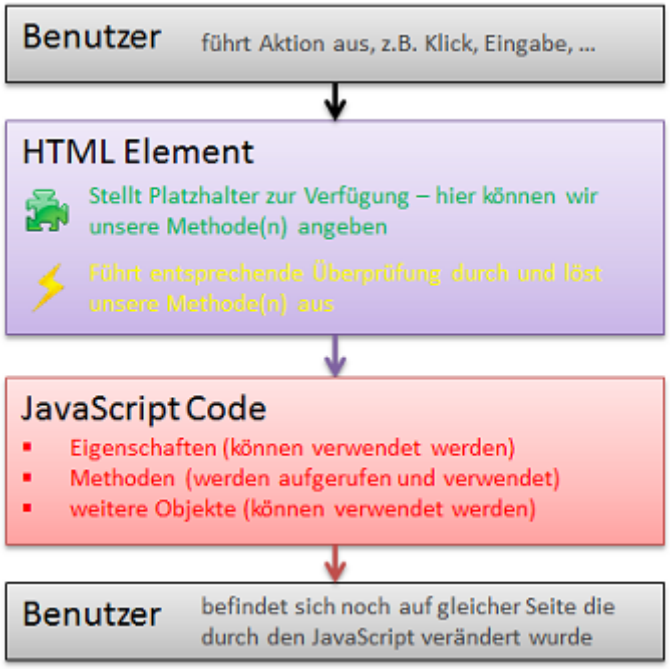
Referenzen:
JS in HTML Tutorial [http://www.dummies.com/how-to/content/javascript-and-html.html]
Mozilla Referenz [https://developer.mozilla.org/en/DOM/document.getElementById]
Tutorial zum Thema getElementById [http://www.tizag.com/javascriptT/javascript-getelementbyid.php]
Historisches zur JavaScript Einbindung [http://devedge-temp.mozilla.org/library/manuals/2000/javascript/1.3/guide/embed.html]
JavaScript in HTML Forms nutzen [http://www.javaworld.com/javaworld/jw-06-1996/jw-06-javascript.html]
Elemente per Namensattribut nutzen [http://www.w3schools.com/jsref/met_doc_getelementsbyname.asp]

Interaktion über Ereignisse

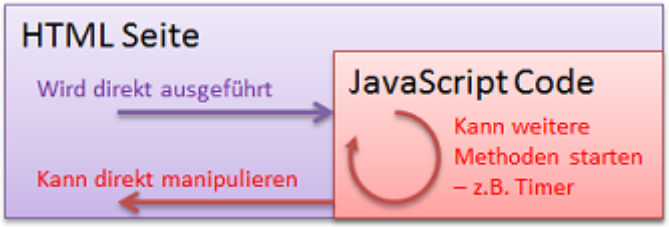
- Ereignisse (Events) sind Aktionen die von JavaScript detektiert werden können
- Jedes HTML Element hat einige bestimmte Ereignisse, die wir nutzen können
- Zunächst definieren wir die Ereignisbehandlung im HTML Tag des entsprechenden Elements
- Beispiele für Ereignisse:
  - Ein Mausklick (`onclick`)
  - Absenden des Formulars (`onsubmit`)

- Eingabe / Änderung von Formelement (onchange)
- Betreten der Seite (`onload`)
- Mit der Maus über ein Element fahren (`onmouseover`)

- JavaScript ist komplett ereignisgesteuert



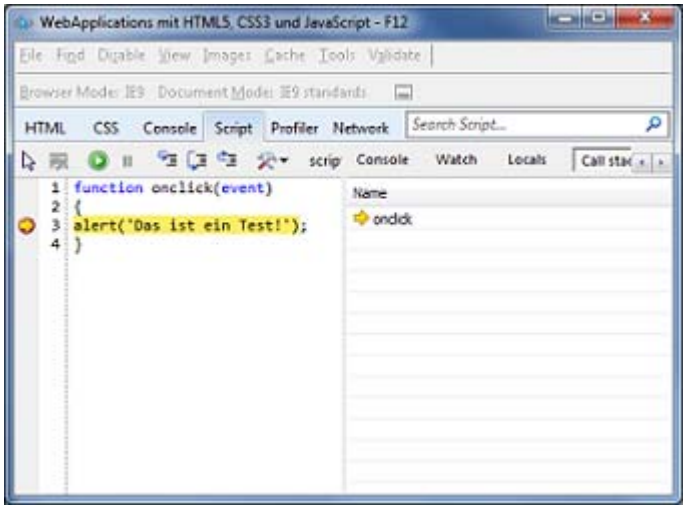
Der Nutzen von Ereignissen liegt in der Interaktion mit dem Benutzer



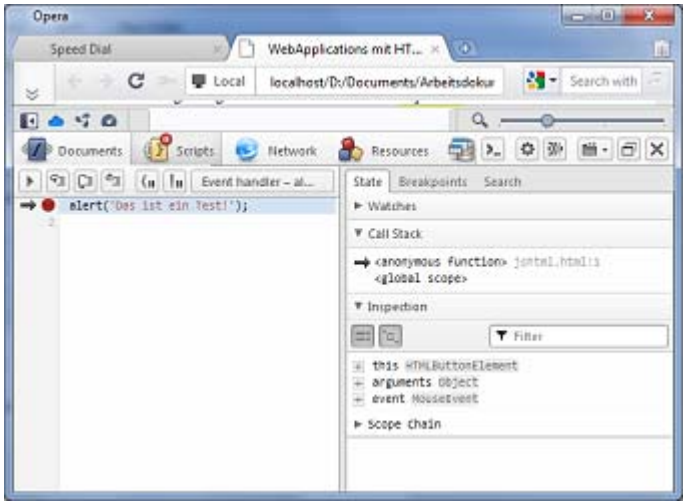
JavaScript benötigt keine Ereignisse zum Ausführen

Referenzen:
W3Schools über Events [http://www.w3schools.com/js/js_events.asp]
Quirksmode über Ereignisse [http://www.quirksmode.org/js/introevents.html]
Ereignisse in JS Tutorial [http://www.comptechdoc.org/independent/web/cgi/javamanual/javaevents.html]

Fehler im Code finden



Verwenden der Entwicklertools im IE9



Der Opera Dragonfly in Aktion

- Fehler finden ist ziemlich schwierig, da es keine Compilerfehler gibt (Skriptsprache)
- Entweder über `alert()` eigene Kontrollen einbauen oder (geschickter) Entwicklertools nutzen
- Im Prinzip hat jeder der großen 5 Browser sehr gute Entwicklertools
- Opera und Chrome sind ziemlich ähnlich - leichte Vorteile für Opera
- IE hat sehr gute JavaScript-Debugger Möglichkeit
- Firebug galt früher als Referenz und ist immer noch gut dabei
- Neben den eingebauten Debuggern kann man noch eigene installieren

- Sehr wichtig ist auch noch der sog. DOM Inspektor

Referenzen:
Erklärung zu Firebug [http://thecodecentral.com/2007/08/01/debug-javascript-with-firebug]
Infos über Debugging Tools [http://qooxdoo.org/documentation/general/debugging_tools]
Liste der eingebauten Debugger [http://skypoetsworld.blogspot.com/2008/01/browser-debugging-tools.html]

Beispiel (3)

JavaScript Code:

```
01. //Setzt die Farbe der Box mit der ID testBox
02. function setColor(e) {
03.     var bg = 'none';//Standardwert
04.     //Schauen ob selectedIndex != 0
05.     if(e.selectedIndex)
06.         bg = e.options[e.selectedIndex].value;
07.     //Wert über das Style Attribut setzen
08.     document.getElementById('testBox').style.background = bg;
09. }
10. //Setzt den Wert des auslösenden Elements
11. function setButtonName(e) {
12.     e.innerHTML = document.getElementById('testBox').style.background;
13. }
```

HTML Code:

```
01. <!--Box erstellen-->
02. <div id="testBox" style="background: none;">
03.     <!--Optionsfeld mit Ereignis erstellen-->
04.     <select onchange="setColor(this);">
05.         <option>---</option>
06.         <option value="#F00">Red</option>
07.         <option value="#0F0">Green</option>
08.         <option value="#00F">Blue</option>
09.     </select>
```

```
10.     <!--Button erstellen und Ereignis festlegen-->
11.     <button onclick="setButtonName(this);">Anklicken</button>
12. </div>
```

- Neues hier: Der Auslöser kann über `this` die eigene Referenz mitgeben
- Eigenschaft eines jeden Elements: `style` – eine von vielen Eigenschaften davon ist `background`

Referenzen:
Die style Eigenschaft [http://www.w3schools.com/jsref/dom_obj_style.asp]
Die options Collection des select Elements [http://www.w3schools.com/jsref/coll_select_options.asp]

DOM Schnittstelle zu HTML

- DOM = **D**ocument **Ob**ject **M**odel
- Nach dem sog. Parsen der HTML-Seite ist der DOM-Tree, der die Struktur der Seite repräsentiert, erstellt
- Die genauen Details interessieren uns nicht – für uns ist wichtig, dass wir durch den DOM einiges können:
    - Auf Elemente des Dokuments zugreifen
    - Elemente zum Dokument hinzufügen (anhängen, einfügen, ...)
    - Elemente des Dokuments entfernen
    - Elemente des Dokuments bearbeiten (z.B. Eigenschaften)
- Die Änderungen am DOM werden vom Browser sofort bemerkt und daher sofort umgesetzt
- Wichtig: Objekte hinzuzufügen ist effizienter als `innerHTML` zu verändern

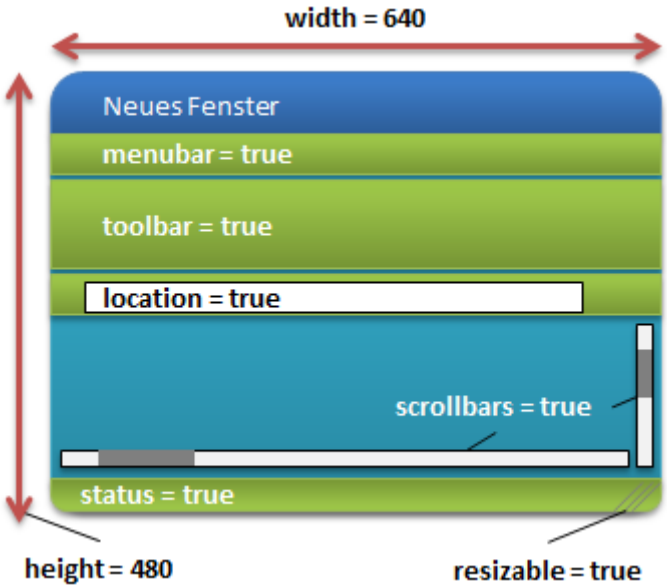
Generierung des DOM durch den Parser (Webkit)

Auswertung des DOM-Trees mit dem IE9

Referenzen:
W3Schools über das DOM [http://www.w3schools.com/htmldom/dom_intro.asp]
Wikipedia Eintrag [http://en.wikipedia.org/wiki/Document_Object_Model]
W3C Spezifikation des DOM (Level 2) [http://www.w3.org/TR/DOM-Level-2-HTML/]
Sehr guter Artikel zum Aufbau von Webbrowsern [http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/]

Das Window Objekt

- Unser `document` liegt im aktuellen `window` Objekt
- Die Ereignisse des Fensters sind ziemlich zentraler Natur, z.B. `onerror` (JavaScript Fehler), `onfocus` (Fenster fokussiert), sowie `onresize` und `onscroll` (Ändern der Fenstereigenschaften)
- Wir können über `history` auch auf den Verlauf des aktuellen Fensters zugreifen
- Dank `frames` gibt es die Möglichkeit auf die (veralteten) frames und iframes zuzugreifen
- Eine Methode des Fensters haben wir schon öfters verwendet: `alert()` - daneben gibt es z.B. noch `confirm()` und `prompt()`
- Über `window.open(url, 'Neues Fenster'[, options])` können wir ein neues Fenster (bzw. Tab) aufmachen, wobei `options` einen String darstellt



Anschauliche Erklärung der möglichen Optionen beim Öffnen eines neuen Fensters mit JavaScript - die Angaben sind als Beispiel zu verstehen, wobei mehrere Angaben durch ein Komma getrennt werden

Referenzen:
Referenz zum Window Objekt [http://www.javascriptkit.com/jsref/window.shtml]
Schnelle Referenz auf devGuru [http://www.devguru.com/technologies/ecmascript/quickref/window.html]
W3Schools Tutorials zum window Objekt [http://www.w3schools.com/jsref/obj_window.asp]
Kochrezepte beim Arbeiten mit dem window Objekt [http://www.perlscriptsjavascripts.com/tutorials/javascript/window_1.html]
Kommunikation zwischen Fenstern in JS [http://sharkysoft.com/tutorials/jsa/content/027.html]

Arbeiten mit Zeichenketten

- Strings (Zeichenketten) sind der am häufigsten benutzte Datentyp, da jede Eingabe zunächst ein String ist
- Strings zu manipulieren (verbinden, entfernen, suchen, ...) gehört daher zu den häufigsten Aufgaben
- JavaScript beherrscht von Grund auf sehr viele Methoden um mit Strings effizient arbeiten zu können
- Die Eigenschaft `length` gibt uns die Länge zurück z.B. ist `'hallo'.length` gleich 5

Einige wichtige Methoden

- `concat()`
- `indexOf()` und `lastIndexOf()`
- `match()`
- `replace()`

- `slice()` und `split()`
- `substr()` und `substring()`

Methoden mit HTML Ausgabe

- `link()`
- `bold()` und `italics()`
- `fixed()`
- `anchor()`
- `small()` und `strike()`
- `sub()` und `sup()`

Referenzen:
Artikel bei Quirksmode [http://www.quirksmode.org/js/strings.html]
W3Schools Artikel über Strings in JS [http://www.w3schools.com/jsref/jsref_obj_string.asp]
Die komplette Referenz zu Strings [http://www.hunlock.com/blogs/The_Complete_Javascript_Strings_Reference]
Die eingebauten Stringmethoden [http://www.tizag.com/javascriptT/javascript-string-functions.php]

Konvertieren von Strings

- Wie erwähnt: Jede Eingabe erstmal ein String - wie wird dieser z.B. zu einer Zahl?
- Von einer Zahl zu einem String ist recht einfach z.B. bei `var i = 2` über var str = i.toString()
- Umgekehrt ist es nicht so einfach - eine Möglichkeit ist `var str = '123'` mit `var i = str * 1`
- Was geschieht wenn wir aber folgendes verwenden: `var str = 'abc'`? Wir erhalten `i = NaN`
- **NaN** heißt *Not a Number* und ist ein spezieller Wert! Funktion zum Test ist `isNaN()`
- Konvertierungsmethoden wie `parseInt()` und `parseFloat()` können auch verwendet werden
- Geben zwar auch im ungünstigen Fall `NaN` zurück - eignen sich aber sehr gut um wirklich Komma- oder Ganzzahl zu erhalten
- Boolsche Konvertierung: alles konvertiert, z.B. ergibt 0 oder auch `null` als Wert `false`
- Arrays mit Elemente, Objekte mit Inhalten etc. ergeben als Wert `true`
- Auswerten von Benutzereingabe: mit regulären Ausdrücken (`match()`) und Typüberprüfung arbeiten

Referenzen:
JS Type Konvertierungen [http://jibbering.com/faq/notes/type-conversion/]
Strings von und zu Zahlen [http://www.javascripter.net/faq/convert2.htm]
Implizite Boolsche Konvertierungen [http://www.bennadel.com/blog/904-Javascript-s-Implicit-Boolean-Conversions.htm]

Beispiel (4)

```
01. var win = window.open('', '', 'width = 400, height = 400, toolbars = false, menubar = false');
02. win.document.write('Link auf meine Homepage'.bold().link('http://www.florian-rappl.de'));//...
03. var qty = win.document.createElement('input');
04. qty.type = 'text';
05. win.document.body.appendChild(qty);//... - noch mehr Code
06. var bt = win.document.createElement('button');//... -noch mehr Elemente erstellen
07. bt.onclick = function() {
08.     var q = parseFloat(qty.value.replace(',','.'));
09.     var f = parseFloat(fak.value.replace(',','.'));
10.     var w = cur.value;
11.     if(isNaN(q) || isNaN(f))
12.         alert('Bitte eine korrekte Zahl eingeben z.B. 5.0!');
13.     else
14.         result.innerHTML = 'Der Betrag ist: ' + (q * f).toString().concat(' ').concat(w).bold();
15. };//Noch mehr Elemente hinzufügen und mehr Abfragen etc.
16. var result = win.document.createElement('p');
17. win.document.body.appendChild(result);
```

- Neu hier: Nutzen das **Chaining** aus (Verkettung) – erhalten String zurück den wir wieder manipulieren
- Ebenfalls neu und vielleicht ungewohnt: sog. **Nesting** von Funktionen

JavaScript Strangeness

- Zunächst folgender Code:

```
01. var a = 0;
02. var b = -0;
03. if(a === b)
04.     alert('Gleich!');
05. if(1/a !== 1/b)
06.     alert('Ungleich!');
```

- Wer hätte das gedacht?! In JavaScript ist 0 gleich 0, **aber** das Vorzeichen wird gespeichert!
- Nun folgender Code:

```
01. var i = 1;//hier wird i_außen gesetzt
02. (function () {
03.     alert(i);//undefined - hier ist schon var i_innen = null;
04.     var i = 2;//jetzt ist i_innen = 2;
05.     alert(i);//2 - wir sind ja im inneren Scope!
06. })();
```

- Das Problem ist, dass wir im inneren Scope `i` neu definieren (Funktionen geben Scope, genannt Kontext)

Referenzen:
Codeproject Artikel [http://www.codeproject.com/KB/scripting/javascript-gotchas.aspx]
Blog Artikel über 9 Gotchas [http://www.fitzblog.com/bid/2127/Nine-Javascript-Gotchas]

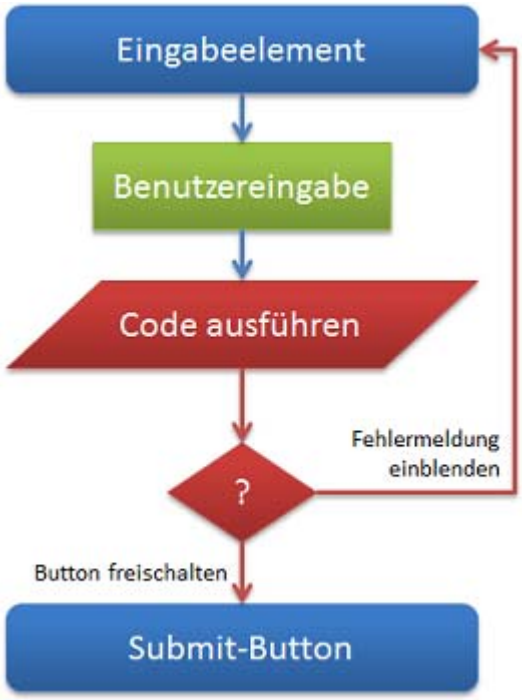
HTML Forms

- Formulare geben uns die Möglichkeit eine Benutzereingabe zu realisieren und diese zu verarbeiten
- Prinzipiell könnte ein Formular auch ohne den `<forms>` Tag aufgebaut werden, allerdings kann dieses dann nicht (ohne JavaScript) an einen Webserver gesendet werden
- In einem Formular sind dann einige `<input>` Elemente wie z.B. TextBoxen (Attribut `type=text`) oder CheckBoxen (`type=checkbox`) uvm.
- Besondere Steuerelemente sind Buttons `<button>`, DropDownBoxen `<select>` und Textfelder `<textarea>`
- Heutzutage würde man gern andere Wege gehen:
  - Statt Daten direkt zu senden und eine neue Seite zu erhalten möchte man das per AJAX (durch JS!) erledigen
  - Vor dem Absenden sollen die Daten (per JS!) bereits überprüft werden um unnötige Requests zu vermeiden
  - Aus mehreren Gründen braucht man aber immer noch eine Lösung mit abgeschalteten JavaScript
- Der große Vorteil in JS liegt in einer interaktiveren Seite mit weniger Requests und einem besseren Erlebnis für Benutzer

Referenzen:
HTML Forms [http://www.w3schools.com/html/html_forms.asp]
Beispiel zu AJAX Forms [http://www.tizag.com/ajaxTutorial/ajaxform.php]
W3C Dokument [http://www.w3.org/TR/html4/interact/forms.html]
Tutorial zu HTML Forms [http://www.tizag.com/htmlT/forms.php]
Wikipedia zu Webforms [http://en.wikipedia.org/wiki/Form_(web)]

Per JavaScript prüfen

- Wie vorhin erwähnt – (lokale) Überprüfung sollte immer durchgeführt werden
- Wichtig: Ohne JavaScript sollte Button auf jeden Fall funktionieren
- Bei aktiviertem JavaScript deaktivieren wir den Button direkt oder führen beim Drücken des Buttons eine Abfrage durch, ob der Submit abgebrochen werden soll
- Für Validierung eignet sich sehr gut der eingebaute RegExp Datentyp
-



Schematischer Ablauf der Validierung

- Code für das Beispiel mit e als Referenz auf die Textbox mit E-Mail Adresse:

```
01. var s = e.value.toLowerCase(), t = document.getElementById('mailtestfehler');
02. t.innerHTML = (/^[a-z0-9][a-z0-9_\.-]{0,}[a-z0-9]@[a-z0-9][a-z0-9_\.-]{0,}[a-z0-9][\.][a-z0-9]{2,4}$/.test(s)) ? '' :
    '<small>Bitte gültige Mail Adresse eingeben!</small>';
```

Referenzen:
Bibliothek mit vielen Regulären Ausdrücken [http://regexlib.com/]
Noch mehr Reguläre Ausdrücke [http://www.regular-expressions.info/]
RegExp bei Wikipedia [http://de.wikipedia.org/wiki/Regulärer_Ausdruck]

Tutorial zu Regulären Ausdrücken [http://www.danielfett.de/internet-und-opensource,artikel,regulaere-ausdruecke]
W3C Schools über Form-Validierung [http://www.w3schools.com/js/js_form_validation.asp]
Form-Validierung schnell und einfach [http://www.javascript-coder.com/html-form/javascript-form-validation.phtml]

Forms in HTML5

- Zunächst: Elemente wie `<button>`, `<input>`, `<select>` und `<textarea>` müssen nun nicht mehr in einem `<form>` Element enthalten sein
- Über das Attribut `form` kann man die `id` (nicht den `name`!) eines zugeordneten `<form>` Elements angeben
- Die neuen Eingabeelemente erhält man über das `type` Attribut für `<input>`:
    - `email` hier wird eine E-Mail Adresse erwartet und `url` zur Eingabe einer Internetadresse
    - `date` ermöglicht die bequeme Auswahl eines Datums und `time` zum Einstellen einer Uhrzeit
    - `datetime` das Kombinationselement zur Datums-und Uhrzeitauswahl
    - `month` Element zur Auswahl eines Monats und `week` Element zur Auswhal einer Woche
    - `color` öffnet einen eingebauten Farbauswahl-Dialog
    - `number` eine Box mit einer Zahlenauswahl und `range` ein Bereichsslider
    - `search` ein explizit benanntes Suchfeld und `tel` ermöglicht Eingabe einer Telefonnummer
- Verwendung des (alten) `<input>`-Tags: Funktioniert als Textbox auf älteren Browsern, leicht zu merken, ...
- Vorteile: Validierung über den Browser ersetzt JavaScripts, spezialisierte Eingabe auf z.B. Smartphones, ...

Referenzen:
Tutorial zu den neuen Elementen [http://24ways.org/2009/have-a-field-day-with-html5-forms]
Unterstützung der neuen Elemente [http://wufoo.com/html5/]
HTML5 Forms sind im kommen [http://snook.ca/archives/html_and_css/html5-forms-are-coming]
Völlig neue Form-Elemente (noch kein Standard) [http://www.w3schools.com/html5/html5_form_elements.asp]
Die neuen Elemente mit Fallback Erklärung [http://www.htmlgoodies.com/html5/tutorials/whats-new-in-html5-forms-email-url-and-telephone-input-types.html#fbid=l1hlkhtIZeL]

Automatische Validierung

- Es gibt einige neue Attribute wie `autofocus`, `placeholder=""`, `required`, `multiple` und `pattern=""`
- Letzteres erlaubt Eingabe eines regulären Ausdrucks: Grund Validierung!
- Einige spezielle Elemente (z.B. `<input type=email>`) haben bereits eine vordefinierte Validierung eingebaut
- Für z.B. **number** gibt es `min`, `max` (Steuerung des Bereichs) und `step` (Steuerung der Schrittweite)
- Neben Validierung können wir über das Attribut `autocomplete` nun steuern ob diese Funktion vom Browser verwendet werden soll
- Problem ist, dass die Validierung erst beim Absenden ausgeführt wird (direkt nur über JavaScript)

- Attribute zum Deaktivieren: `novalidate` (ein Element) oder `formnovalidate` (alle Elemente) im Button

Anzeige einiger HTML5 Eingabefelder in Opera mit automatischer Validierung

Referenzen:
Grundsätzliches zur HTML5 Validierung [http://www.broken-links.com/2011/03/28/html5-form-validation/]
Ein wenig mehr Tiefgang [http://www.the-art-of-web.com/html/html5-form-validation/]
Opera über Validierung [http://dev.opera.com/articles/view/improve-your-forms-using-html5/]
Präsentation über HTML5 Validierung [http://www.slideshare.net/ianoxley/html5-form-validation-7731661]
Fallback für ältere Browser [http://www.webresourcesdepot.com/cross-browser-html5-form-validation-jquery-html5form/]

DIY Validierung

- Wir wollen nun die automatische Validierung nutzen um eine eigene Validierung einzubauen
- Dies ist sinnvoll, da die automatische Validierung erst beim `onsubmit` sichtbar wird

Der entsprechende JavaScript und HTML Code:

```
01. function validateEmail(e) {//unsere Methode - willValidate ist ebenfalls neu und zeigt uns ob es HTML5 ist
02.     if(e.willValidate && !e.validity.valid) {//Jedes Element besitzt (in HTML5) eine validity Eigenschaft
03.         document.getElementById('mailerror').innerHTML = 'Bitte eine korrekte Mailadresse eingeben';
04.         document.getElementById('formmailsubmit').disabled = true;//Button disablen
05.     } else if (e.value.length) {
06.         document.getElementById('mailerror').innerHTML = ' ';
07.         document.getElementById('formmailsubmit').disabled = false;//Button enablen
08.     }
09. }
```

```
01. <!-- Einige Angaben zum Formular und entsprechende Labels -->
02. <input type="email" placeholder="Bitte E-Mail 'name@domain.com' eingeben" onkeyup="validateEmail(this);" />
03. <!-- Weiteres inkl. Submit Button mit entsprechender ID -->
```

Referenzen:
Diskussion über HTML5 Validierung [http://stackoverflow.com/questions/432933/will-html-5-validation-be-worth-the-candle]

E-Mail Adressen mit HTML5 Validieren [http://email.about.com/od/emailprogrammingtips/qt/How-To-Validate-Email-Addresses-With-Html5.htm]
Tutorial zur Validierung mit HTML5 (inkl. `setCustomValidity()`) [http://www.script-tutorials.com/data-validation-html5/]

Auf Eingaben reagieren

- Entweder über onchange, onkeyup (ein Element) etc. oder neu über `onforminput` (alles) im Form
- Oder beim Versenden des Formulars über das Ereignis `onsubmit` im entsprechenden `<form>` Element
- Haben über den Rückgabewert der Methode die Möglichkeit den Versand abzubrechen (`return false`)
- Ein kleines Beispiel:

  Name*
  Telefon*
  Anmerkung

  Abschicken (* Pflichtfeld)

- Der verwendete HTML und JavaScript Code:

```
01. <form ... onsubmit="return validateForm();" ... name="vform">
02. <!-- ... und noch mehr code -->
03. </form>
```

```
01. function validateForm() {
02.     var f = document.vform;
03.     var error = [];//Array mit Fehlern
04.     if(f.formname.value.length == 0)
05.         error.push('...');
06.     if(f.formtel.value.length == 0)
07.         error.push('...');
08.     else if(!/^\([1-9][0-9]{2}\)\s?[0-9]{4}(\-[0-9]+)?/.test(f.formtel.value))
09.         error.push('...');
10.     document.getElementById('formerror').innerHTML = error.join('<br/>');
11.     return !error.length;//=> 0->true else->false
12. }//bei Rückgabe von false kein Submit
```

Referenzen:
Frage bzgl. Abbrechen eines Submits [http://bytes.com/topic/javascript/answers/494532-how-do-i-cancel-submit]
Auf Tastatureingaben reagieren [http://www.javascriptkit.com/javatutors/javascriptkey.shtml]
O'Reilly über Forms mit JS [http://docstore.mik.ua/orelly/webprog/jscript/ch15_01.htm]
Tutorial des Grinnell Colleges [http://www.math.grin.edu/~rebelsky/Tutorials/JavaScript/EdMedia97/forms.html]

Die jQuery Bibliothek

Mit der Verwendung von jQuery entsteht eine Abhängigkeit

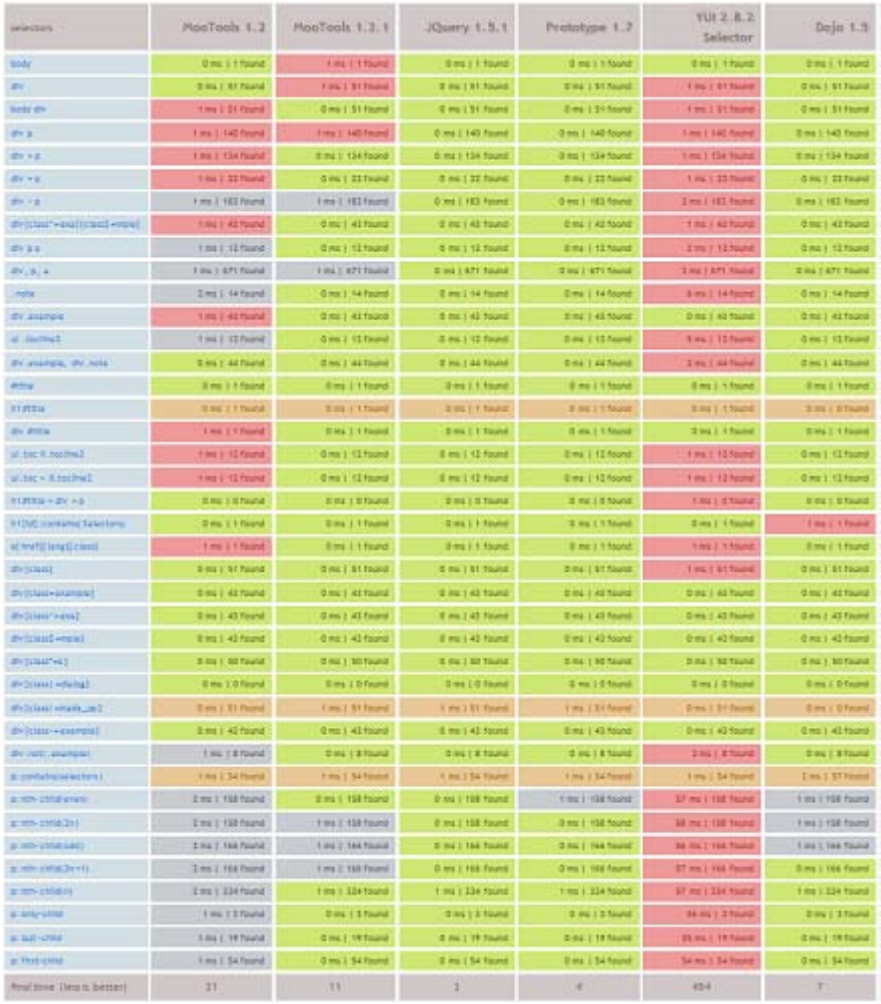
- Eine der größten JavaScript Bibliotheken mit der weitesten Verbreitung
- Erleichtert den DOM Zugriff: Manipulation und Auswertung
- Cross-Browser und Legacy-Browser fähig (jede Methode berücksichtigt Browsertyp)
- Jede Menge Plugins (zusätzlicher JavaScript Code der jQuery verwendet bzw. erweitert) vorhanden

Der Komponentenweise Aufbau der jQuery Bibliothek

Referenzen:

Die jQuery Bibliothek [http://jquery.com]
Alle jQuery Projekte [http://jquery.org/]
Tutorial über jQuery [http://www.w3schools.com/jquery/default.asp]
Wikipedia Artikel [http://en.wikipedia.org/wiki/JQuery]

Warum jQuery?



jQuery (ältere Version) gewinnt mit nur 3 ms sehr leicht (YUI 484 ms)

- CSS ähnliche Syntax – JavaScript für Designer
- Unzählige Plugins erweitern das Grundpaket
- Microsoft unterstützt jQuery sehr stark (Debugging)
- Es werden die Elemente gefunden die gesucht werden
- Chaining erleichtert die Arbeit ungemein
- Die Geschwindigkeit ist ungeschlagen
- Es ist sehr klein (Kleinere Größe = Schnellere Ladezeit!)
- Sehr leicht erweiterbar

- Es wird u.a. bei Google gehostet - extern auf sehr gutem Server verwendbar

Referenzen:
Vergleichstest auf der MooTools Seite [http://mootools.net/slickspeed/]
Diskussion bei StackOverflow [http://stackoverflow.com/questions/67045/what-advantages-does-jquery-have-over-other-javascript-libraries]
Vorteile durch Verwendung von jQuery [http://www.jquery4u.com/articles/benefits-jquery/]

Elemente auswählen

- Einbinden von jQuery erfolgt durch `<script>` Tag[1] (Download von jQuery.com)
- Mit jQuery wird die Auswahl eines HTML-Elements sehr viel einfacher
- Die Auswahl erfolgt durch einen CSS ähnlichem Syntax (Vorteil für später)
- Wir können (0-n) Elemente direkt, per ID, per Klassen oder Pseudoklassen (z.B. `:hover`) ansprechen, z.B.
    - Ansprechen per ID:

```
01. document.getElementById('foo');//Alter Weg
02. $('#foo');//Neuer Weg
```

    - Ansprechen per Klasse:

```
01. document.getElementsByClassName('foo');//Alter Weg (wir müssten noch [0] schreiben bzw. durchiterieren)
02. $('.foo'); //Neuer Weg (könnten nur die 0 auswählen ansonsten gelten Änderungen für alle!)
```

      oder Tag-Name:

```
01. document.getElementsByTagName('input');//Alter Weg siehe getElementsByClassName
02. $('input'); //Neuer Weg analog zu $('.foo')
```

Referenzen:
[1] Es empfiehlt sich jQuery von Google zu laden, da man hier Traffic spart und immer die aktuelle oder gewünschte Version erhält
jQuery bei Google [http://code.google.com/apis/libraries/devguide.html#jquery]
jQuery Selectors [http://api.jquery.com/category/selectors/]
Seite zum rumspielen mit Selectors [http://codylindley.com/jqueryselectors/]
Tutorial zu jQuery Selectors [http://www.netmagazine.com/tutorials/getting-most-out-jquery-selectors]

Durch Arrays mit jQuery

- jQuery hilft uns nicht nur bei der Arbeit mit dem DOM sondern auch mit weiteren JS-Objekten wie Arrays

- Durch die Methode `$.each(liste, callback)` können wir auf Listen zugreifen, z.B.

```
01. var a = ['Eins', 'zwei', 'Drei'];//Array mit drei Einträgen
02. $.each(a, function(i, v) { alert('Der Index ist ' + i + ' und der Wert ist: ' + v); });
```

- Der **Callback** kann also zwei Parameter enthalten: Index und Wert des aktuellen Elements
- Vorteil in der Verwendung von `each()`: Wir können direkt durch jQuery Objekte durchlaufen, z.B.:

```
01. $('.someClass').each(function(i, v) { //geht durch jeden gefundenen Eintrag mit der Klasse someClass
02.     alert('Der Index ist ' + i + ' und die Höhe ist: ' + $(v).height());
03. });
```

- Der Wert v in der Callback-Funktion gibt uns das DOM Objekt - machen durch `$(v)` ein jQuery Objekt
- Was ist eigentlich $? Dies ist die Kurzschreibweise für `jQuery`[1]!
- Nochmal zum *Chaining* - jQuery Regel: Keine sinnvolle Rückgabe => gibt jQuery Instanz zurück

Referenzen:

[1] Dies kann z.B. über `var j = jQuery.noConflict();` auf `j` geändert werden
jQuery No Conflict [http://api.jquery.com/jQuery.noConflict/]
each auf ein jQuery Object [http://api.jquery.com/each/]
each auf eine Collection [http://api.jquery.com/jQuery.each/]
Informationen zum Konstruktor [http://api.jquery.com/jQuery/]
Das Tolle an jQuery.each [http://www.bennadel.com/blog/534-The-Beauty-Of-The-jQuery-Each-Method.htm]
Fünf Beispiele zu each [http://www.jquery4u.com/jquery-functions/jquery-each-examples/]

Elemente hinzufügen

- Wir erinnern uns noch an den langen Syntax von zuvor (`createElement(...)` uvm. notwendig)
- jQuery schafft hier Abhilfe z.B. könnten wir sehr einfach ein Element hinzufügen:

```
01. $('<p/>').text('Wenn ich hier HTML schreibe z.B. <p> Dann kommt &lt;p&gt; raus!').appendTo('#testBoxjQueryAdd');
02. $('<p/>').html('<strong>Hier ist HTML was wert!</strong>').appendTo('#testBoxjQueryAdd');
```

- Testen wir das gleich mal: Text-Beispiel | HTML-Beispiel
- jQuery erstellt also nicht vorhandene Elemente direkt - wir können auch sofort deren Eigenschaften (`css()`, `attr()`, `height()`, `width()`, `text()`, `html()`, ...) bearbeiten
- Hinzufügen zu einem Element müssen wir sie allerdings (immer) noch - hier helfen uns aber `appendTo()`, `prependTo()`, `append()` und `prepend()`

weiter

- Auch das Entfernen geht sehr viel leichter - es gibt `remove()`, `detach()`, `empty()` und `unwrap()`

Referenzen:
jQuery Elemente hinzufügen [http://api.jquery.com/category/manipulation/dom-insertion-inside/]
jQuery Methode appendTo() [http://api.jquery.com/appendTo/]
HTML über html() einfügen [http://api.jquery.com/html/]
Text über text() einfügen [http://api.jquery.com/text/]
Tutorial wie jQuery funktioniert [http://docs.jquery.com/How_jQuery_Works]
Quick-Guide zum Chaining [http://tobiasahlin.com/blog/quick-guide-chaining-in-jquery/]

Elemente ändern

- Einer der Vorteile im Verwenden von jQuery liegt in der (universalen) Syntax zum Ändern von Elementen
- Im Prinzip gelten folgende Regeln:
  - Will man das Attribut eines HTML Tags ändern (oder auslesen) so verwendet man `attr()`, z.B.

```
01. var getter = $('#beispielID').attr('href');//Speichert den Wert des Attributes href in getter
02. $('#beispielID').attr('href', 'beispiel.html');//Setzt den Wert (und gibt jQuery-Instanz zurück)
```

  - Will man den Style eines HTML Elements ändern (oder auslesen) so verwendet man `css()`, z.B.

```
01. var getter = $('#beispielID').css('display');//Speichert den Wert der Eigenschaft display des Styles
02. $('#beispielID').css({ 'display' : 'none'});//So kann man auch mehrere Werte setzen - Objekt!
```

  - Will man das Ereignis eines HTML Elements ändern (oder auslösen) so verwendet man `bind()`, z.B.

```
01. $('#beispielID').bind('click');//Führt das Click-Ereignis aus
02. $('#beispielID').click();//Sehr viel kürzere Alternative - speziell für Click
03. $('#beispielID').bind('click', function() {});//Setzt das Click-Ereignis
04. $('#beispielID').click(function() {});//Ebenfalls kürzere Alternative
```

Referenzen:
jQuery Events [http://api.jquery.com/category/events/]
jQuery EventHandler binden [http://www.barneyb.com/barneyblog/2009/04/10/jquery-bind-data/]
jQuery zur DOM Manipulation [http://api.jquery.com/category/manipulation/]
10 nützliche Codeschnippsel zu DOM-Veränderungen [http://www.catswhocode.com/blog/manipulating-the-dom-with-jquery-10-useful-code-snippets]
6 nützliche Codes für Formular-Veränderungen [http://calisza.wordpress.com/2009/03/29/6-jquery-snippets-you-can-use-to-manipulate-select-inputs/]

Tutorial um mit jQuery das DOM zu verändern [http://www.tutorialspoint.com/jquery/jquery-dom.htm]

Mit jQuery prüfen

- jQuery bietet uns einige Möglichkeiten um Forms leichter und effizienter zu validieren
- *Beispiel 1* Das jQuery Plugin **Validation**
- Wie funktioniert das mit den Plugins?
    - Nach Einbinden von jQuery binden wir noch eine weitere JavaScript-Datei (das Plugin) ein
    - Durch das Plugin gibt es eine bis mehrere neue Methoden bei jQuery bzw. jQuery Objekten
    - In diesem Fall sind die Methoden `validate()`, `rules()` und `valid()` neu
    - Einstellung über CSS-Klassen wie `required`, `email`, uvm. (Vorteil: Konkretes Styling möglich)
- *Beispiel 2* Selbst durch Ausnutzen des `data-` Attributes
- Was ist das `data-` Attribut und wie kann man es verwenden?
    - Möglichkeit eigene Daten in DOM Objekten (bzw. HTML Code) zu hinterlegen
    - Könnten z.B. die Attribute `data-pattern` und `data-required` erstellen (falls passend)
    - Beim Laden der Seite führen wir dann `$('input[data-required=true]').each(...)` aus
    - Setzen für jedes gefundene Element das onchange Ereignis mit dem entsprechenden Pattern

Referenzen:
Dokumentation des jQuery Plugins [http://docs.jquery.com/Plugins/validation]
Prüfen ob ein Element ein Attribut hat [http://api.jquery.com/has-attribute-selector/]
John Resig über das `data-` Attribut [http://ejohn.org/blog/html-5-data-attributes/]
HTML5 Doktor über eigene Attribute [http://html5doctor.com/html5-custom-data-attributes/]
Die jQuery Methode `attr()` [http://api.jquery.com/attr/]
Tutorial über jQuery Form Validierung [http://speckyboy.com/2009/12/17/10-useful-jquery-form-validation-techniques-and-tutorials-2/]

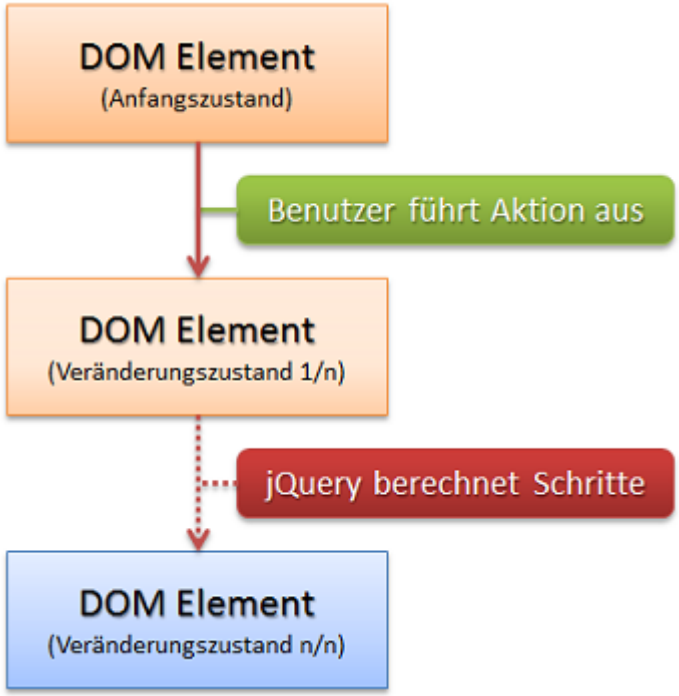
Wann jQuery ausführen?

- Im Prinzip können wir z.B. das `onload`-Ereignis des `<body>`-Tags verwenden
- Problem: Das DOM ist zu dieser Zeit u.u. noch nicht fertig und wir erhalten Fehler
- Des weiteren ist es ungünstig JavaScript im `<head>` einzubinden - immer erst am Schluss vor `</body>`
- Daher ist jQuery noch gar nicht eingebunden und `$` bzw. `jQuery` existiert nicht
- Lösung: Verwenden der `ready()` Methode. Hier gibt es zwei mögliche Wege mit
    - `$(document).ready(function() {...})` bzw.
    - `$(function() {...})`, was wesentlich kürzer aber unanschaulicher ist.

- Vorteile gegenüber dem `onload`-Ereignis:
  - DOM ist auf jeden Fall fertig und alles, was benötigt wird, wurde geladen
  - Die Methode kann für mehrere Skripte verwendet werden (nacheinander ausgeführt)
  - Kann auch nach dem Laden ausgeführt werden – wird dann instantan ausgeführt
  - Garantiert also Ausführung und zwar zum günstigsten Zeitpunkt

Referenzen:
jQuery zur Ready Methode [http://api.jquery.com/ready/]
Das Load Ereignis mit jQuery [http://api.jquery.com/load-event/]
Das Unload Ereignis mit jQuery [http://api.jquery.com/unload/]
Tutorial zur Ready Methode [http://docs.jquery.com/Tutorials:Introducing_$(document).ready()]

Vitalität dank Animationen



Animationen mit jQuery laufen iterativ ab

- jQuery bietet uns sehr viele Möglichkeiten Animationen zu gestalten
- Neben vordefinierten Animationen über `slideUp()`, `slideDown()` und `toggleSlide()` etc. kann man auch sehr schnell eigene Animationen erstellen
- Die entsprechende Methode heißt `animate()`

- `stop()` stoppt die aktuelle Animation des Elements
- Beispiel:

```
01. function inAni(e) {
02.     $(this).stop().animate({height : '20px', width : '100px'}, 2000, function() { ... });
03. } //Setzen den Text ready über Callback
04. function outAni(e) {
05.     $(this).stop().text('').animate({height : '50px', width : '50px'}, 500);
06. } //Entfernen den Text vor der Animation
```

Referenzen:
jQuery Animationen [http://api.jquery.com/animate/]
Eigene Effekte in jQuery [http://api.jquery.com/category/effects/custom-effects/]
Vordefinierte Fading Effekte [http://api.jquery.com/category/effects/fading/]
Vordefinierte Sliding Effekte [http://api.jquery.com/category/effects/sliding/]
jQuery Animationen in 5 Minuten [http://addyosmani.com/blog/jquery-sprite-animation/]

JavaScript Timers

- Wie genau funktioniert das mit den n Frames bei den Animationen? Durch Timers! Grob: T / 13 = n
- T ist hier die angegebene Zeit in *ms* durch 13 *ms / frame* ist die Zeit pro Frame (Standard in jQuery[1])
- Welche Möglichkeiten haben wir?
  - `setInterval()` zum Starten einer Schleife und `clearInterval()` zum Stoppen

```
01. var ms; //Variable zum Merken der Intervallschleife
02. function start() { if(!ms) ms = setInterval(move, 100); }//Startet das Intervall falls nicht schon gestartet (mit einem
    Callback und der Zeit in ms)
03. function stop() { clearInterval(ms); ms = null; }//Stoppt das Intervall, Reset von ms
04. function move() { $('#testbox').css('margin-left', "+=2px"); }//Bewegt die Box (alle 100ms) um 2px
```

  - `setTimeout()` zum Starten einer Einmalverarbeitung und `clearTimeout()` zum Stoppen

```
01. var t = 11;//10 Sekunden (dekrementieren vor der Verzweigung)
02. $(timer);//Soll sofort (mit jQuery.ready()) starten
03. function timer() {
04.     $('#jstimerex').text('noch ' + (--t) + ' s');//Ausgabe
```

```
05.     if(t) setTimeout(timer, 1000); else alert('Timer abgelaufen!'); }//Erneut oder Alert?
```

Referenzen:

[1] jQuery mit 77 fps [http://forum.jquery.com/topic/why-jquery-uses-77-fps-by-default-in-animation]
Änderung der Standard fps [http://api.jquery.com/jQuery.fx.interval/]
W3Schools über Timers [http://www.w3schools.com/js/js_timing.asp]
W3Schools über Interval [http://www.w3schools.com/jsref/met_win_setinterval.asp]
Tutorial zu Timers [http://www.elated.com/articles/javascript-timers-with-settimeout-and-setinterval/]
Beispiel zu `setInterval()` [http://www.pagecolumn.com/javascript/setinterval.htm]

Ereignisse mit jQuery

- Einer der größten Vorteile in der Verwendung von jQuery: Ereignisse
- jQuery verwendet selbst Ereignisse in Form von Callbacks an vielen Stellen (z.B. Animationen)
- Daneben können wir noch unsere Ereignisse sehr bequem anbinden (`bind()`) oder entfernen (`unbind()`)
- Um alle Events mit einem Handler abzuhören kann man `live()` verwenden
- Über `die()` kann man mit einer Methode alle EventHandler eines Elements entfernen
- Einer der größten Vorteile in der Verwendung von Events mit jQuery sind die (**Event**-)Parameter
- Beispiel eines MouseMove Ereignisses:

```
01. $('#testtarget').mousemove(function(ev) {
02.   var pos = $(ev.target).offset();
03.   var msg = 'EventHandler für .mousemove() PAGE { x = ' + ev.pageX + ', y = ' + ev.pageY + '} INNR { x = ';
04.   msg += (ev.pageX - pos.left) + ', y = ' + (ev.pageY - pos.top) + ' } TIME { ' + ev.timeStamp + ' }';
05.   $('#testlog').text(msg); });
```

Referenzen:
Events mit jQuery [http://api.jquery.com/category/events/]
jQuery Hilfe zu bind [http://api.jquery.com/bind/]
jQuery Hilfe zu unbind [http://api.jquery.com/unbind/]
jQuery Hilfe zu die [http://api.jquery.com/die/]
jQuery Hilfe zu live [http://api.jquery.com/live/]
Globale in lokale Koordinaten umwandeln [http://www.bennadel.com/blog/1871-Translating-Global-jQuery-Event-Coordinates-To-A-Local-Context.htm]

Was passiert ohne JavaScript

- Bisher sind wir immer davon ausgegangen, dass entweder HTML5 und/oder JavaScript beim Benutzer vorhanden ist

- Dies ist in der Realität nicht (immer) der Fall – HTML5 wird derzeit als High-End Technologie betrachtet
- Zusätzlich kann JavaScript manchmal deaktiviert oder nicht vorhanden sein – welche Möglichkeiten besitzen wir?
- Empfohlen wird die Seite so zu designen, dass Sie auch ohne JavaScript akzeptabel aussieht
- Mit JavaScript führen wir die (gewollten) Änderungen beim Start der Seite ($(. . .)) aus
- Wollen wir zusätzlich noch Inhalt anzeigen so bietet sich eine HTML-Lösung an:

```
01. <!--... Quellcode der Seite-->
02. <noscript>
03. JavaScript ist deaktiviert: Bitte JavaScript aktivieren um weitere Features nutzen zu können!
04. </noscript>
05. <!--... Quellcode der Seite-->
```

- Die Lösung über <noscript> ist elegant und schon länger implementiert

Referenzen:
Der NoScript Tag auf W3Schools [http://www.w3schools.com/tags/tag_noscript.asp]
HTMLDog Beispiel zu NoScript [http://www.htmldog.com/reference/htmltags/noscript/]
Sitepoint Erklärung und Details [http://reference.sitepoint.com/html/noscript]

Nesting im Visier

- Unter Nesting verstehen wir im Prinzip nichts anderes als Verschachteln, z.B. ist bekannt

```
01. if (value > 5.0) { //äußerste If-Anweisung
02.     if (value > 25.0) { //mittlere If-Anweisung
03.         if (value > 100) { /* ... */ } //innerste If-Anweisung
04.         else { /* ... */ } //else der innersten
05.     } else if (value > 10) { /* ... */ //else if der mittleren
06.     } else { /* ... */ } //else der mittleren
07. } else { /* ... */ } //else der äußersten
```

- In JavaScript können wir aber auch Methoden in Methoden schachteln
- Dies bietet uns einige Vorteile (Zugriff auf Kontextvariablen), hat allerdings Performance Nachteil
- Methoden sind notwendig um einen Kontext für Variablen zu erzeugen, z.B.

```
01. var a = ['Erster Eintrag','Zweiter Eintrag','Dritter Eintrag','Vierter Eintrag'];
```

```
02. for(var i = 0; i < a.length; i++) {
03.     $('<a />').text(a[i]).appendTo('body').css({ display : 'block'}).click(function() { alert(i); });
```

wird hier das `i` von außen referenziert, d.h. wir erhalten immer 4 zurück

Referenzen:
Artikel zum sinnvollen Nesten [http://net.tutsplus.com/tutorials/javascript-ajax/stop-nesting-functions-but-not-all-of-them/]
If-Anweisungen verschachteln [http://javascript.about.com/od/decisionmaking/a/des09.htm]
Mehrdimensionale Arrays durch Nesting [http://www.elated.com/articles/nested-arrays-in-javascript/]
JavaScript Methoden verschachteln [http://www.devarticles.com/c/a/JavaScript/Nesting-JavaScript-Functions/]
Beispiel zur Gefahr beim Nesten [http://html5.florian-rappl.de/nesting.html]

Beispiel (5)

JavaScript Code:

```
01. var el = null;//Explizites NULL setzen
02. $('#selection').change(function (e) {
03.     var s = this;
04.     if(!s.selectedIndex || el) return;
05.     el = s.innerHTML;//Speichern Originalzustand
06.     $('<button/>').text('Zurück').click(function(){
07.         $(this).remove();//Nesting
08.         $(s).html(el);
09.         el = null;
10.     }).appendTo($(s).parent());//anhängen
11.     var val = parseInt(s.selectedIndex);
12.     var values = [val * 1, val * 2, val * 3];
13.     $(s).html('');//innerHTML reset
14.     $.each(values, function(i,v) {//Nesting
15.         $(s).append('<option>' + v + '</option>');
16.     });//Elemente hinzufügen
17. });
```

HTML Code:

```
01. <!--Erste Auswahlliste mit JS-Ereignis-->
```

```
<select  id="selection">
<!--Option für NICHTS (---)-->
    <option>---</option>
    <option>1</option>
    <option>2</option>
    <option>3</option>
</select>
<!--Was passiert ohne Script?!-->
<noscript>
<!--Geben 2. Auswahlliste-->
    <select>
        <option>1</option>
        <option>2</option>
        <option>3</option>
    </select>
</noscript>
```

"Klassen" in JavaScript

- In JavaScript gibt es kein Schlüsselwort um Klassen direkt zu erzeugen (Tricksen ist jedoch erlaubt!)
- Jedoch bietet uns JS drei Möglichkeiten eine Klasse zu erstellen
    - Als Objekt (hier haben wir aber keinen Konstruktor und daher nur eine Instanz verwenden) über

```
var NameDerKlasse = {};//Erstellen des Object-Literals
NameDerKlasse.WertDerEigenschaft = 5;//Eigenschaft
NameDerKlasse.FunktionsName = function() {...};//Methode
```

    - Als anonyme Funktion mit dem new Operator (hier haben wir einen sinnlosen Konstruktor)

```
var NameDerKlasse = new function(...) { //Erstellen und zuweisen der anonymen Methode
    this.WertDerEigenschaft = 5;//Eigenschaft
    this.FunktionsName = function() {...}; };//Methode
```

    - Als benannte Funktion mit dem new Operator (hier haben wir einen sinnvollen Konstruktor)

```
01. function UnsereKlasse(...) { //Erstellen der benannten Methode
02.     this.WertDerEigenschaft = 5;//Eigenschaft
03.     this.FunktionsName = function() {...}; };//Methode
04. var NameDerKlasse = new UnsereKlasse(...);//erstellt Instanz!
```

Referenzen:
Erstellen einer Klasse in JS [http://webdevelopersjournal.com/articles/jsintro3/js_begin3.html]
OOP in JS [http://www.codeproject.com/KB/aspnet/JsOOP1.aspx]
OOP Klassen erstellen [http://www.webmonkey.com/2010/02/make_oop_classes_in_javascript/]
Drei Wege um Klassen zu definieren [http://www.phpied.com/3-ways-to-define-a-javascript-class/]
Konstruktor Trick [http://james.padolsey.com/javascript/terse-javascript-101-part-1/]

Private und Public

- Eben haben wir das Schlüsselwort `this` verwendet um Eigenschaften und Funktionen festzulegen (=**public**)
- Was würde bei Verwendung von `var` passieren? Wir könnten (außerhalb) nicht darauf zugreifen (=**private**)
- Das bedeutet, dass wir in JavaScript zwischen (int.) Variablen und (ext.) Eigenschaften unterscheiden können

```
01. function Rectangle(x, y, w, h) {
02.     this.X = x || 0; this.Y = y || 0; //Setzen über den Konstruktor die Werte - x || 0 ist Trick um null Wert abzufangen
03.     this.Width = w || 0; this.Height = h || 0; //Das selbe mit diesen Werten (alles PUBLIC)
04.     var moves = 0; //PRIVATE Variable
05.     this.Move = function(dx, dy) { //Funktion ist public und steuert interne
06.         moves++;//Moves Variable [ nur intern! ]
07.         alert('Wurde bereits ' + moves + ' mal bewegt!'); this.X += dx || 0; this.Y += dy || 0; };
08.     this.ShowMessage = function() { //Zeigt die Daten an (auch hier muss this verwendet werden)
09.         alert('Pos: ' + this.X + ', ' + this.Y + '; Dim: ' + this.Width + ', ' + this.Height); };
10. }
```

Referenzen:
Elemente zur Select-Liste hinzufügen [http://www.mredkj.com/tutorials/tutorial005.html]
Private Variablen in JavaScript [http://javascript.crockford.com/private.html]
Erstellen einer Klasse in JS [http://webdevelopersjournal.com/articles/jsintro3/js_begin3.html]
OOP in JS [http://www.codeproject.com/KB/aspnet/JsOOP1.aspx]

Methoden in Klassen

- Natürlich ist wie bis jetzt gezeigt eine Definition in der Klasse (Nesting) möglich
- Dies ist aber leider nicht sehr günstig, da die Methoden mit jeder Instanz erneut erstellt werden
- Wir können Methoden auch außerhalb deklarieren und innerhalb verwenden, z.B.

```
01. function UnsereKlasse(...) {//...
02.     this.FunktionsName = foo; }//Verweis auf Methode die außerhalb liegt
03. function foo() { ... };//Hier ist foo deklariert
```

- Dies ist schneller, aber leider auch von außerhalb ohne eine Instanz verwendbar (ungünstig!)
- Am elegantesten und schnellsten ist daher eine Lösung über die `prototype` Eigenschaft, z.B.

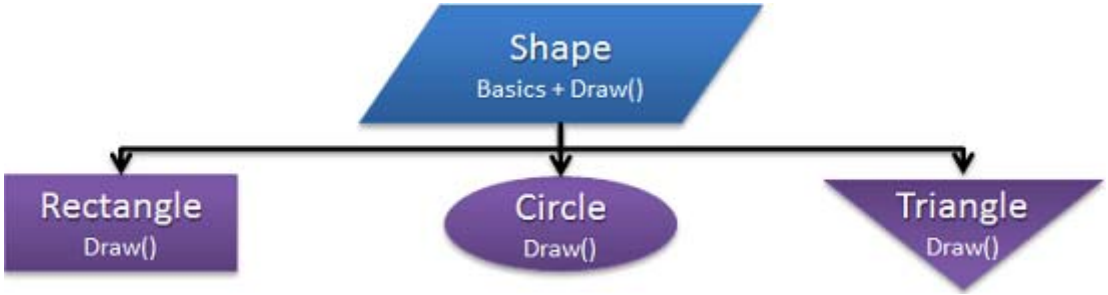
```
01. function UnsereKlasse(...) { /* ... */ }//Alle interna werden zugewiesen
02. UnsereKlasse.prototype.FunktionsName = function() { ... };//Alle externa werden zugewiesen
```

- Der `prototype` ist für einige OOP Eigenschaften in JavaScript zuständig
- Problem bei diesen Ansätzen: Wir können nur auf Eigenschaften (`this.`) zugreifen

Referenzen:
Objects erstellen [http://www.howtocreate.co.uk/tutorials/javascript/objects]
Erstellen einer Klasse in JS [http://webdevelopersjournal.com/articles/jsintro3/js_begin3.html]
OOP in JS [http://www.codeproject.com/KB/aspnet/JsOOP1.aspx]

Polymorphie in JavaScript

- Polymorphie bedeutet, dass man eigene Implementierungen bekannter Methoden erstellen kann



Beispiel einer Vererbung – Rectangle, Circle und Triangle erben von Shape

- Können über das `prototype` Schlüsselwort die Basis einer Klasse definieren
- Um dem Beispiel aus der Grafik zu folgen:

```
01. function Shape() { //Grundsätzliches festlegen, hier x und y Startpunkt
02.     this.X = 0; this.Y = 0; } //Wie immer public Eigenschaften mit this
03. Shape.prototype.Draw = function () { ... };//Draw von Shape als Standard festlegen
04. Rectangle.prototype = new Shape();//Vererbung erfolgt hier (Basis ist ein neuer Shape)
05. Rectangle.prototype.Draw = function () { ... };//Eigene Implementierung (zeichnet Rechteck)
06. Circle.prototype = new Shape();//Vererbung erfolgt hier (Basis ist ein neuer Shape)
07. Circle.prototype.Draw = function () { ... };//Eigene Implementierung (zeichnet Ellipse)
08. Triangle.prototype = new Shape();//Vererbung erfolgt hier (Basis ist ein neuer Shape)
09. Triangle.prototype.Draw = function () { ... };//Eigene Implementierung (zeichnet Dreieck)
```

Referenzen:
Artikel über OOP und JS [http://mckoss.com/jscript/object.htm]
Vererbung in JavaScript [http://www.zipcon.net/~swhite/docs/computers/languages/object_oriented_JS/inheritance.html]
OOP in JS [http://www.codeproject.com/KB/aspnet/JsOOP1.aspx]
Polymorphe JavaScript Methoden [http://javascript.about.com/library/blpolyfunc.htm]
Wikipedia Artikel zu OOP [http://en.wikipedia.org/wiki/Object-oriented_programming]

Namensräume und statische Felder

- Namensräume verfolgen das Ziel Objekte voneinander abzugrenzen um Ordnung zu schaffen
- Wir können dies in JavaScript sehr leicht schaffen indem wir unsere Klassen in Objekte schachteln, z.B.

```
01. var Namensraum = { };//Einen Namensraum erstellen
02. Namensraum.Shape = function() { ... };//Den Konstruktor im Namensraum erstellen
03. Namensraum.Shape.prototype.Draw = function() { ... };//Funktionen hinzufügen
04. var instanz = new Namensraum.Shape();//Instanz erstellen
```

- Statische Eigenschaften und Methoden (Felder) können ebenfalls hinzugefügt werden
- Hier gehen wir analog zu den `prototypes` vor – nur ohne dieses Schlüsselwort, d.h.

```
01. Shape = function() { ... };//Den Konstruktor wie vorhin erstellen
02. Shape.Foo = function() { ... };//Statische Methode foo angelegt
03. Shape.Value = 10;//Statische Eigenschft Value an- und festgelegt
```

- Statische Felder sind instanzlos (d.h. unabhängig) und können von jeder Instanz aufgerufen werden

- Statische Methoden sehen nur weitere statische bzw. im aktuellen Kontext vorhandene Felder

Referenzen:
Statische Eigenschaften und Methoden [http://elegantcode.com/2011/01/19/basic-javascript-part-7-static-properties-and-methods/]
Namensräume in JavaScript [http://elegantcode.com/2011/01/26/basic-javascript-part-8-namespaces/]
Statische Felder erstellen [http://justjs.org/js-oop/static-methods-in-javascript/]

Beispiel (6)

```
01. var Auto = function(dist) {            //Konstruktor
02.     Auto.Count++;                       /*Erhöhen die Zahl aller Autos*/
03.     this.Distance = dist || 0;           //Legen Start km Fest
04.     var isDriving = false, velocity = 0.0, timeStarted = null, id = Auto.Count;
05.     this.GetIsDriving = function() { return isDriving; };    //Machen einen Getter draus
06.     this.GetVelocity = function() { return velocity; };    //Machen einen Getter draus
07.     this.GetId = function() { return id; };    //Machen einen Getter draus
08.     this.SetVelocity = function(v) {    //Brauchen interne Methode wg. privaten Variablen
09.         var currentTime = new Date().getTime();    //Aktuelle Uhrzeit auslesen
10.         if(isDriving)        //Wenn er schon fährt sollen die vorhandenen km addiert werden
11.             this.Distance += velocity * Math.floor((currentTime - timeStarted) / 1000);
12.         if(isDriving = !!v)        //setze aktuellen Zustand und wenn er jetzt fährt dann
13.             timeStarted = currentTime;        //setzen wir die aktuelle Zeit
14.         velocity = v; }; }
15. Auto.Count = 0;            //Statisches Feld - Speichert Zahl aller Autos
16. Auto.prototype.StartDriving = function(velocity) { //externer Prototype
17.     this.SetVelocity(velocity); };        /*Setzt die Geschwindigkeit*/
18. Auto.prototype.toString = function() {
19.     return 'Auto ' + this.GetId() + '/' + Auto.Count + '\nDistanz: ' + this.Distance + '\nFährt gerade: ' + (this.GetIsDriving() ?
    'Ja' : 'Nein') + '\nGeschwindigkeit ' + this.GetVelocity() + ' km/h';
```

Das Audio Element

- HTML5 erlaubt uns Audiodateien direkt über den <audio>-Tag einzubinden
- Der Tag ist von den Attributen her wie der <img>-Tag aufgebaut
- Das Element ist aus Fallbackgründen nicht selbstschließend – der Inhalt ist ein Fallback, z.B.

```
01. <audio src="...">Ihr Browser unterstützt noch kein HTML5 Audio. Bitte <a href="...">hier</a> klicken zum Download.</audio>
```

- Natürlich gibt es neben `src` jede Menge neuer Attribute für dieses Element, u.a.
  - `autoplay` spielt die Audiodatei automatisch ab (Vorsicht!)
  - `controls` zeigt Kontrollelemente (Start, Pause, ...) an (ansonsten gar nichts!)
  - `loop` stellt ein ob es sich um eine Endlosschleife handelt (Standardmäßig: `false`)
  - `preload` mit drei Werten (`auto`, `none`, `metadata`) zur Konfiguration des Vorladevorgangs
- Neue JavaScript-Ereignisse sind u.a. `play`, `canplay`, `pause`, `seeking`, `abort`, `ended` und `error`
- Wir können viele neue Eigenschaften (z.B. `volume`) bearbeiten und Methoden (z.B. `play()`) aufrufen

Referenzen:
Liste der unterstützten Formate [http://www.w3schools.com/html5/html5_audio.asp]
W3Schools über den Audio-Tag [http://www.w3schools.com/html5/tag_audio.asp]
W3C Spezifikation des Audio-Tags [http://dev.w3.org/html5/spec/Overview.html#the-audio-element]
HTML5 Doctor über den Audio-Tag [http://html5doctor.com/native-audio-in-the-browser/]
jQuery Audioplayer für HTML5 [http://jplayer.org/]

Das Video Element

- Analog zum Audio-Element, besitzt jedoch zusätzlich noch Videobildanzeige
- Zusätzliche Attribute sind unter anderem gegeben durch
  - `poster` um ein Vorschaubild zu spezifizieren (ansonsten wird der erste Frame verwendet)
  - `height` bzw. `width` um die Größe des Videos zu bestimmen
- Schauen wir uns doch mal ein Beispiel mit eigenen Steuerelementen an (von hier):

```
01. <div class="videochrome paused">
02.     <div class="controls">
03.         <div class="scrub">
04.             <button class="play" title="play">&#x25BA;</button>
05.             <div class="duration">0:00</div>
06.             <div class="loaded"><div class="buffer"><div class="playhead"><span>0:00</span></div></div></div>
07.         </div>
08.     </div>
09.     <video loadeddata="canplay();" play="playEvent();" pause="pausedEvent();" ended="this.pause();"
```

```
        progress="updateSeekable();" durationchanged="updateSeekable();" timeupdate="updatePlayhead();" src="leverage-a-synergy.ogv">
        </video>
    </div>
```

Referenzen:
Seite über HTML5 Videos [http://html5video.org/]
Wikipedia Artikel zu HTML5 Video [http://en.wikipedia.org/wiki/HTML5_video]
HTML5Rocks Artikel zum Video Tag [http://www.html5rocks.com/en/tutorials/video/basics/]
W3Schools über die Videoformate [http://www.w3schools.com/html5/html5_video.asp]
W3C Spezifikation des Video Elements [http://www.w3.org/TR/html5/video.html]
SublimeVideo - HTML5 Video Player mit Fallback-Lösung [http://sublimevideo.net/]
Ausführlicher Artikel über den video-Tag [http://www.chipwreck.de/blog/2010/03/01/html-5-video-dom-attributes-and-events/]

Beispiel (7)

```
var video = document.getElementsByTagName('video'),//Holen uns das 1. Video-Element
    duration = $('.duration').get(0),//Holen uns 1. class=duration Element
    play = $('.play'),//Holen uns das Element mit class='play'
    scrubWidth, //Hier wird die Breite des Scrubbings eingestellt
    currentTime = $('span').get(0);//Holen uns das 1. Span-Element
function canplay() { //Wird aufgerufen sobald abgespielt werden können
  this.height = this.videoHeight; this.width = this.videoWidth;
  //Legen die Größe des angezeigten Videos fest - Größe soll gleich dem echten Video sein
  this.parentNode.style.height = this.videoHeight + 'px';  this.parentNode.style.width = this.videoWidth + 'px';
  //Legen die Größe des Scrubbings fest (hier soll der Benutzer die Position des Videotracks einstellen können)
  scrubWidth = parseFloat(getComputedStyle(buffer.parentNode, null).width);
}
function playEvent() {
  $(this.parentNode).removeClass('paused'); play.attr('paused', true); //Klasse entfernen und Attribut hinzufügen
}
function pausedEvent() {
   $(this.parentNode).addClass('paused'); play.attr('paused',''); //Klasse hinzufügen und Attribut entfernen
}
play.click(function () { //Code wird beim Klicken auf den Play-Button ausgeführt
    if (video.ended) //Sollte das Video zuende sein spulen wir zurück
```

```
21.         video.currentTime = 0; //Das machen wir über das Setzen der aktuellen Zeitmarke auf 0
22.     if (video.paused) video.play() else video.pause(); //Ansonsten Pausieren oder Abspielen
23. });
```

Das Format-Desaster

- Zwar konnten sich die Hersteller auf einen Audio- und Video-Tag einigen, jedoch nicht auf einen einheitlichen Standard
- Daher akzeptieren diese Tags noch einen Untergeordneten Tag - den <source>-Tag, z.B.

```
01. <video controls>
02.     <source src="leverage-a-synergy.ogg" type="video/ogg; codecs='theora, vorbis'">
03.     <source src="leverage-a-synergy.mp4" type="video/mp4; codecs='avc1.42E01E, mp4a.40.2'">
04.     <embed src="http://www.youtube.com/v/cmtcc94Tv3A&h1=en_GB&fs=1&rel=0" type="application/x-shockwave-flash"
    allowscriptaccess="always" allowfullscreen="true" width="425" height="344">
05. </video>
```

- Hier verwenden wir einen öffentlichen Codec (OGG) und einen kommerziellen Codec (MPEG-4)
- Eine Fallback Lösung über Adobe Flash ist ebenfalls integriert

					
OGG*	nein	ja	ja	ja	nein
AAC(*)	ja	nein	nein	ja	ja
MP3	ja	nein	nein	ja	ja
WAV	nein	ja	ja	ja	ja

Die Unterstützung der wichtigsten Audio-Codecs

					
OGG*	nein	ja	ja	ja	nein
webM*	nein	ja	ja	ja	nein
MPEG-4	ja	nein	nein	ja	ja
H.264	ja	nein	nein	ja	ja

Die Unterstützung der wichtigsten Video-Codecs

Mit * gekennzeichnete Formate sind **frei erhältlich**.

Beim Verwenden eines frei erhältlichen und eines kostenpflichten Codecs wird jeder Browser unterstützt.

Referenzen:
Das Original Beispiel auf Seite zum Buch [http://introducinghtml5.com/examples/ch04/ch4-full-fallback.html]
Wieviele Formate sind notwendig? [http://blog.zencoder.com/2010/10/06/how-many-formats-do-i-need-for-html5-video/]
Vergleich der einzelnen Codecs [http://webdesign.about.com/od/video/a/html5-video-formats.htm]
Über AAC und OGG [http://www.scirra.com/blog/44/on-html5-audio-formats-aac-and-ogg]
Das webM-Projekt (FAQ) [http://www.webmproject.org/about/faq/]

Grafik mit Canvas

- Audio und Video sind zwar schön, aber noch lange kein Grund Flash bzw. Silverlight abzulösen
- Was HTML (schon immer) gefehlt hat ist die Möglichkeit **frei** darauf zu zeichnen
- Mit dem Canvas-Tag erhalten wir nun die Möglichkeit in JavaScript darauf zeichnen zu können

Mit dem Canvas Element sind dynamische Zeichnungen auf eine Website möglich

- Wie bei den anderen neuen Media-Tags hat auch <canvas> einen Fallback durch den Inhalt
- Die einzigen eigenen Attribute sind `width` und `height` zum Einstellen der Größe in Pixel

Referenzen:
W3Schools über den Canvas-Tag [http://www.w3schools.com/html5/tag_canvas.asp]
[http://www.w3schools.com/html5/html5_canvas.asp]
[http://en.wikipedia.org/wiki/Canvas_element]
MDN Tutorials zu Canvas [https://developer.mozilla.org/en/Canvas_tutorial]
W3C Spezifikation des Canvas Elements [http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html]
Jede Menge Tutorials unter HTML5CanvasTutorials.com [http://www.html5canvastutorials.com/]
Einführender Artikel bei Codeproject [http://www.codeproject.com/KB/HTML/html5-canvas-part1.aspx]

Canvas JavaScript Methoden

- `getContext('2d')` **gibt** uns das entsprechende Zeichenobjekt (hier: 2D, z.B. auch möglich: **webgl**)
- Das 2D-Kontextobjekt hat einige Eigenschaften, z.B.
  - `fillStyle` (Füllfarbe), `strokeStyle` (Stiftfarbe) und `lineWidth` (Stiftbreite)
  - `shadowOffsetX/Y`, (Schattenversatz), `shadowBlur` (Unschärfe) und `shadowColor` (Schattenfarbe)
  - `font` (Schriftart), `textAlign` (hor. Ausrichtung) und `textBaseline` (ver. Ausrichtung)
- Neben diesen Eigenschaften gibt es noch jede Menge Methoden (mit eigenen Objekten), z.B.
  - `fillRect(x,y,width,height)` (Rechteck füllen), `strokeRect(x,y,width,height)` (Rechteck zeichnen) und `clearRect(x,y,width,height)` (Rechteck mit Hintergrundfarbe füllen)
  - `beginPath()` und `closePath()` um Pfad zu erstellen, der durch `fill()`, `stroke()` gefüllt wird
  - Mit `moveTo(x,y)` bzw. `lineTo(x,y)` werden Teilstücke des Pfades festgelegt
  - Über `drawImage(img_el,x,y,width,heigth)` können wir Bilder einbinden
  - Durch `createImageData(x,y,width,height)` kann Pixelkarte ein Bereich mit Pixelmanipulation erstellt werden
- Sehr gute Einsatzmöglichkeiten des Canvas-Tags bieten sich durch ein Video-Element

Referenzen:
Opera Canvas Tutorial [http://dev.opera.com/articles/view/html-5-canvas-the-basics/#the2dcontextapi]
WebGL Tutorial [http://wiki.delphigl.com/index.php/Tutorial_WebGL]
MDN über Canvas [https://developer.mozilla.org/en/Canvas_tutorial]
W3Schools Canvas Tutorial [http://www.w3schools.com/html5/html5_canvas.asp]

Beispiel (8)

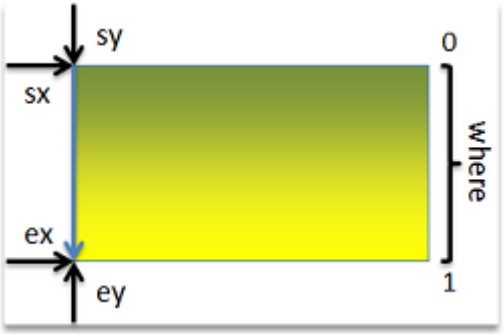
```
01. var mouseDown = false, point = {}, //Variablen zum Speichern des vorherigen Punktes und des Maustastenzustands
02.     canvas = document.getElementById('canvas'); //Das Canvas ELement geholt
03. canvas.onmousemove = moveHandler; //Den Movehandler festlegen (Maus wird bewegt)
04. canvas.onmouseup = function () { mouseDown = false; }; //Den Uphandler festlegen (Maustaste losgelassen)
05. canvas.onmousedown = downHandler; //Den Downhandler festlegen (Maustaste gedrückt)
06. var c = canvas.getContext('2d'); //Das entsprechende Context-Objekt holen
07. c.strokeStyle = '#666666'; c.lineWidth = 1; //Grundlegende Eigenschaften festlegen
08. function downHandler(ev) {
09.     mouseDown = true; //Maustastenzustand speichern
```

```
10.     point.x = ev.clientX; point.y = ev.clientY; //Diesen Punkt als vorherigen Punkt speichern
11. }
12. function moveHandler(ev) {
13.     if(mouseDown) { //Nur wenn die Maustaste gedrückt ist zeichnen wir
14.         c.beginPath(); //Und zwar einen individuellen Pfad
15.         c.moveTo(point.x, point.y); //Vom alten Punkt
16.         c.lineTo(ev.clientX, ev.clientY); //Zum neuen Punkt
17.         c.stroke(); //Zeichnen
18.         c.closePath(); //Pfad schließen
19.         point.x = ev.clientX; point.y = ev.clientY;    } //Neuen Punkt als vorherigen speichern
20. }
```

Dieses Beispiel kann hier ausgeführt werden.

Noch mehr Canvas!

- Mit Canvas können wir auch sehr leicht Gradienten zeichnen (z.B. linearer Farbverlauf)
- Dazu benötigen wir nur die Methode `createLinearGradient(sx,sy,ex,ey)`, die uns ein Objekt gibt
- Dieses Gradient Objekt besitzt die Methode `addColorStop(where,color)`



Die Bedeutung der Argumente für den linearen Farbverlauf

- Der `fillStyle` kann mit diesem Objekt gesetzt werden und mit `fillRect()` gefüllt werden
- Gradangaben beim radialen Farbverlauf sind stets in **rad** (`rad = Math.PI / 180`)
- Über die Canvas-Methode `createPattern(img,mode)` können Muster erzeugt werden (z.B. `'repeat'`)
- Aus jedem Canvas kann über die Methode `toDataURL('image/png')` ein Bild erzeugt werden

Referenzen:
Gradient Tutorial [http://www.tutorialspoint.com/html5/canvas_create_gradients.htm]
Transformationen und Gradienten [http://net.tutsplus.com/tutorials/javascript-ajax/canvas-from-scratch-transformations-and-gradients/]
Sehr cooles Canvas Demo [http://html5demos.com/canvas-grad]
MDN Tutorial zu Canvas [https://developer.mozilla.org/en/Canvas_tutorial/Applying_styles_and_colors]
Stackoverflow zu toDataURL [http://stackoverflow.com/questions/934012/get-image-data-in-javascript]
MSDN zu Canvas toDataURL im IE9 [http://msdn.microsoft.com/en-us/library/ff975241(v=vs.85).aspx]

Bilder laden und verwenden

- Wie wir bereits festgestellt haben können wir mit Canvas auch Bilder zeichen - woher kommen diese?
- Die Bilder müssen durch einen <img>-Tag bereits geladen sein, dies erreichen wir durch

```
01. var img = document.createElement('img');//Erstellt das Bild
02. img.onload = function() { /* Was tun wenn geladen?*/ };//Falls Bild gecached vor src
03. img.src = url;//Legt fest woher das Bild kommt - hier könnten wir auch toDataURL() verwenden
```

- Da wir das Bild nirgends angehängt haben wird es auch nicht auf der Seite angezeigt werden
- Es kann wichtig sein das `onload` Ereignis vor dem Setzen des `src` Attributes festzulegen
- Als Bild erwartet der Canvas-Kontext immer das entsprechende HTML-Image-Element (d.h. den Tag)
- D.h. wir können nun die Variable *img* verwenden z.B. in `createPattern(img, 'repeat')`

Referenzen:
Bild Randomizer [http://www.random-image.net/]
Mit Canvas malen [http://thinkvitamin.com/code/how-to-draw-with-html-5-canvas/]
Bild Manipulationen in Canvas [http://virtuelvis.com/archives/2005/12/canvas-image-manipulation]

Animationen zeichnen

- In Canvas kann man sehr gut Animationen einbauen - dies geht über löschen und neuzeichnen
- Zum Zeichnen könnte man z.B. immer dieselbe Methoden mit unterschiedlichen Argumenten verwenden
- Oder man transformiert die Koordinaten und verwendet dieselbe Methode mit gleichen Argumenten
- Zusätzlich zur Transformation (`translate(dx,dy)` oder `rotate(alpha)`) kann man Zustände speichern
- `save()` speichert den aktuellen Zustand[1] und `restore()` stellt den gespeicherten Zustand wieder her

```
01. var c = document.getElementById('canvas').getContext('2d'), alpha = 0;
02. setInterval(rotate, 100);//Alle 100 ms rotieren
```

```
03. function rotate() {
04.     c.clearRect(0,0,c.canvas.width,c.canvas.height);//Clear
05.     c.save();//Aktuellen Zustand speichern
06.     alpha += 1 / 180 * Math.PI;//Winkel erhöhen
07.     c.translate(c.canvas.width / 2, c.canvas.height / 2);
08.     c.rotate(alpha);/*Rotation geht immer um 0,0 - daher Translate!*/
09.     drawMaze();//Zeichnet unser schönes Maze (einfacher Alg.!)
10.     c.restore();//Vorherigen Zustand wiederherstellen
11. }
```

Referenzen:

[1] Bezieht sich auf `fillStyle`, `font`, `lineWidth`, `strokeStyle`, `textAlign`, uvm...
Einen Kreis zeichnen [http://billmill.org/static/canvastutorial/ball.html]
Einfache Animationen mit dem Canvas Element [http://html5.litten.com/simple-animation-in-the-html5-canvas-element/]
Grundlegendes zu Animationen [https://developer.mozilla.org/en/Canvas_tutorial/Basic_animations]
Canvas Animations-Kit [http://glimr.rubyforge.org/cake/canvas.html#KeyboardTest]
8 sehr coole Animationen [http://www.queness.com/post/3885/8-simply-amazing-html5-canvas-and-javascript-animations]

Beispiel (9)

```
01. for(var i = 0; i < 500; i++) //Sterne generieren
02.     stars.push({x : Math.floor(Math.random() * canvas.width * 2), y : Math.floor(Math.random() * canvas.height)});
03. ballImg.onload = function() { //Erstellen ein <img>-Element namens ballImg
04.     var ball = document.createElement('canvas'); ball.width = 64; ball.height = 64;
05.     b = ball.getContext('2d'); b.translate(32, 32); //Wollen immer genau die Mitte nehmen
06.     setInterval(draw, 30); //Alle 30 ms neu zeichnen über draw Methode
07.  };
08. ballImg.src = 'img/ball.png'; //schönes Ball Bild
09. function draw() {
10.     c.clearRect(0, 0, c.canvas.width, c.canvas.height);    c.save(); //löschen und Zustand speichern
11.     c.translate(-x / 2, 0);
12.     //Sterne zeichnen - einfach durch Array gehen und an stars[i].x und stars[i].y zeichnen
13.     b.rotate(Math.PI / 180 * 5 * dx / Math.abs(dx)); //Den Ball rotieren lassen - in dx Richtung
14.     b.drawImage(ballImg, 0, 0, ballImg.width, ballImg.height, -32, -32, 64, 64);//Ball zeichnen
15.     c.drawImage(b.canvas, x, y);//Zeichnung übertragen
```

```
16.     //Hier steht Ballphysik (x + dx, y + dy, mit PBC Check!) und restore
17. }
```

Video und Canvas

- Eine super Möglichkeit ergibt sich bei der Kombination Video-Tag und Canvas-Element
- Analog zum `<img>`-Element können wir Videos auf der Seite abspielen ohne diese anzuzeigen
- Um den aktuellen Frame zu zeichnen müssen wir nur ein `video` statt `img` übergeben
- Leider funktioniert beim `<video>`-Element das `onload` Ereignis nicht so wie vorhin, daher:

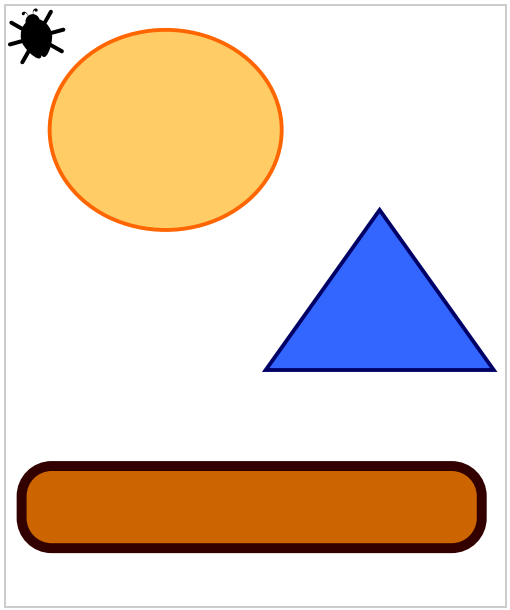
```
01. var jumpTimer = setInterval(function() { //Daten im entsprechenden Videoelement 'video' setzen
02.     try { //Verhindert Werfen eines Fehlers wenn Daten noch nicht bereit sind, der zum Abbruch führt
03.         video.currentTime = start; //Setzt die Startzeit (von wo wollen wir capturen?)
04.         clearInterval(jumpTimer); //Anscheinend erfolgreich - daher nicht mehr ausführen
05.         video.play(); //Abspielen - von Startzeit 'start' aus
06.     } catch(e) {} //Nicht erfolgreich - nach 100 ms wird dies erneut aufgerufen
07. }, 100);
```

- Einige Eventhandler des Videoelements helfen uns beim Festhalten von Bildern:
  - Über `timeupdate` können wir Änderungen der Zeit abfragen
  - Über `loadeddata` sollten wir unsere Schleife starten
- Wichtig ist hier v.a. dass wir über `video.volume = 0` auf Lautlos stellen!

Referenzen:
Beispiel aus Introducing HTML5 [http://introducinghtml5.com/examples/ch05/drawimage.html]
HTML5 Doctor über die Canvas zu Video Verbindung [http://html5doctor.com/video-canvas-magic/]
Das Zerstörungs-Video-Experiment [http://www.craftymind.com/factory/html5video/CanvasVideo.html]
Die Besten Canvas-Video-Demos [http://www.reelseo.com/awesome-html5-canvas-video-demos/]

Vektorgrafiken mit SVG

- Neben pixelbasierten Grafiken mit Canvas kann man auch vektorbasierte Grafiken mit `<svg>` erstellen
- Was mit SVG auf keinen Fall geht sind Pixelmanipulationen (Ereignisse z.B. Klick auf Kreis sind möglich)
- Ein Vorteil von SVG ist, dass man komplizierte Pfade direkt in Programmen wie Inkscape[1] zeichnen kann



- Das Bild wurde nur über HTML generiert (geht bei Opera nicht[2])
- Gezeichnet wurde es in einem externen Programm[3]
- Der generierte Quellcode wurde nur kopiert und wäre über JavaScript noch sehr leicht manipulierbar
- Ein Beispiel dafür: Rechteck Ellipse Dreieck
- Im Prinzip muss man nur neue (SVG-) HTML-Elemente dem <svg>-Element hinzufügen
- Das alles funktioniert daher sehr leicht, da SVG ein XML-basiertes Format darstellt

Referenzen:

[1]Das kostenlose Vektorgrafikprogramm Inkscape [http://inkscape.org/]

[2]Opera kann SVG - aber nur aus externen SVG Dateien - diese können z.B. über *<embed>* oder *<img>* eingebunden werden
StackOverflow über die SVG Einbindung in Opera [http://stackoverflow.com/questions/882272/embed-svg-into-html-opera-specific]

[3] SVG Tool zur Online-Erstellung [http://www.amaltas.org/svgapp/]
Wikipedia Artikel über SVG [http://en.wikipedia.org/wiki/Scalable_Vector_Graphics]
W3C SVG Arbeitsgruppe [http://www.w3.org/Graphics/SVG/]
W3Schools SVG Tutorial [http://www.w3schools.com/svg/default.asp]
Tutorialspoint Artikel [http://www.tutorialspoint.com/html5/html5_svg.htm]
Sehr detaillierter Einstieg [http://www.i-programmer.info/programming/graphics-and-imaging/2063-getting-started-with-svg-html5.html]

Canvas vs. SVG

- SVG bietet daher einige Vorteile gegenüber Canvas:
    - Die Grafiken skalieren sehr schön, d.h. ohne aufzupixeln (sprich: Skalieren und nicht zoomen!)
    - Jedes Objekt (Linie, Dreieck, ...) ist ein echtes Objekt: es können Ereignisse ausgenutzt werden
    - Wir können Objekte direkt manipulieren bzw. löschen ohne dass wir alles neuzeichnen müssen
- Grob gesagt: Für skalierbare Mausinteraktion ist SVG besser, für direkte Tastaturinteraktion Canvas
- Den besten Effekt erhält man natürlich bei Kombination der beiden Techniken
- Canvas gilt eher als Technik für Entwickler, SVG eher für Designer

Einsatzbereiche von Canvas und SVG schematisch aufgelistet

Referenzen:
W3C Blog über Canvas und SVG [http://www.w3.org/QA/2010/05/truly_w3c_community_building_a_2.html]
Opera über die Wahl zwischen Canvas und SVG [http://dev.opera.com/articles/view/svg-or-canvas-choosing-between-the-two/]
Die Qual der Wahl [http://www.sitepoint.com/canvas-vs-svg-how-to-choose/]
StackOverflow über Canvas gegen SVG [http://stackoverflow.com/questions/568136/svg-vs-canvas-where-is-the-web-world-going-towards]
Performance auf dem iPad [http://techblog.floorplanner.com/2011/03/07/canvas-vs-svg-performance-on-ipad/]
Patrick Denglers Artikel zu Canvas vs. SVG [http://blogs.msdn.com/b/ie/archive/2011/04/22/thoughts-on-when-to-use-canvas-and-svg.aspx]

Charts mit SVG

- SVG eignet sich ausgezeichnet um damit einfache (vielleicht sogar dynamische) Charts zu zeichnen
- Ein einfaches dynamisches Beispiel ist unten gezeigt (das Koordinatensystem ist statisch)
- Wichtig ist den Namensraum (`...w3.org/2000/svg`) zu beachten - SVG ist spezialisiert

Wichtiger HTML Code:

```
01. <svg xmlns="http://www.w3.org/2000/svg">
02.     <line x1="10" y1="10" x2="10" y2="90" stroke="#444" />
```

```
    <line x1="10" y1="90" x2="780" y2="90" stroke="#444" />
    <polyline points="10,10 7,15 13,15" fill="#444" />
    <polyline points="780,90 775,87 775,93" fill="#444" />
</svg>
```

Wichtiger JavaScript Code:

```
//Hinzufügen des Elements und setzen der Attribute
var r = document.createElementNS('http://www.w3.org/2000/svg','rect');
document.getElementById('svgChartEx').appendChild(r);
r.setAttribute('x', 10);
r.setAttribute('y', 50);
r.setAttribute('width', 100);
r.setAttribute('height', 40);
r.setAttribute('fill', '#F00');
```

Referenzen:
setAttribute Methode [http://reference.sitepoint.com/javascript/Element/setAttribute]
W3C zum SVG Textelement [http://www.w3.org/TR/SVG/text.html#TextElement]
SVG Textmanipulation in JS [http://bytes.com/topic/javascript/answers/893927-svg-text-manipulation-javascript]
SVG mit JavaScript [http://apike.ca/prog_svg_jsanim.html]
W3Schools JavaScript Beispiele [http://www.w3schools.com/svg/svg_examples.asp]

Grundlegendes zu SVG

- Alle Elemente (z.B. ein Kreis) sind echte HTML-Unterelemente von <svg>
- Die bekanntesten Elemente sind:
  - `circle` mit den Attributen `cx`, `cy` und `r`
  - `ellipse` mit den Attributen `cx`, `cy`, `rx` und `ry`
  - `rect` mit den Attributen `x`, `y`, `width` und `height`
  - `line` mit den Attributen `x1`, `y1`, `x2` und `y2`
  - `polygon` bzw. `polyline` mit dem Attribut `points` (Wert entspricht "x1,y1 x2,y2 ...")
- Generische Attribute sind z.B. `fill`, `fill-opacity` `stroke`, `stroke-width`, `stroke-opacity` etc.
- Wir müssten also zum Erstellen eines SVG-Bildes über JavaScript jede Menge Elemente erstellen und Eigenschaften setzen

- Zur Manipulation müssten wir über unsere aus jQuery oder einfachem JavaScript bekannten Selektoren verwenden
- Alternativ: Write once use anywhere! Wir schauen uns eine mögliche Umsetzung in Form von **RaphaëlJS** an

Referenzen:
Grundlegendes Tutorial [http://www.tutorialspoint.com/html5/html5_svg.htm]
SVG über JavaScript manipulieren [http://www.carto.net/svg/manipulating_svg_with_dom_ecmascript/]

Verwenden von RaphaëlJS

- Es existiert Lösung für SVG mit jQuery - wollen jedoch eine mächtigere unabhängige Lösung
- Der grundlegende Syntax ist allerdings sehr jQuery ähnlich: Konstruktor `Raphael(x,y,width,height)`
- Die damit erhaltene Variable kann Objekte wie z.B. einen Kreis mit `circle(x,y,r)` erstellen
- Diese Methoden geben uns die Objekte direkt zurück (Chaining), so dass wir z.B. `attr(name,value)` verwenden können
- Analog zu jQuery gibt es auch hier `animate()`, `hide()` und `show()`, sowie Events wie `click()` etc.
- Welche weiteren wichtigen Objekte haben wir direkt zur Verfügung?
    - Über `rect(x,y,w,h)` können wir ein Rechteck (x,y ist links oben) erstellen
    - Über `ellipse(x,y,rx,ry)` können wir eine Ellipse (x,y ist Mitte) erstellen
    - Mit `path(p)` kann man einen Pfad erstellen
- Das Argument der letzten Methode sieht z.B. so aus: `M 250 250 l 0 -50 z`
- **M** (Move) startet den Pfad (setzt Startpunkt), **z** schließt den Pfad und **l** (line) bewegt den Pfad (relativ)

Referenzen:
RaphaëlJS [http://raphaeljs.com/]
RaphaëlJS Referenz [http://raphaeljs.com/reference.html]
Freie (SVG) Icons durch RaphaëlJS [http://raphaeljs.com/icons/]
Direktes rumspielen mit RaephaëlJS [http://raphaeljs.com/playground.html]
Tutorial zur Bibliothek [http://net.tutsplus.com/tutorials/javascript-ajax/an-introduction-to-the-raphael-js-library/]
Eine Karte (U.K.) erstellen [http://return-true.com/2011/06/using-raphaeljs-to-create-a-map/]

Beispiel (10)

```
01. var paper = Raphael(10, 50, 320, 200);
02. var rectangle = paper.rect(5,5,100,20);
03. rectangle.NewAttributes = { x: 495, width: 300, fill: '#A03' };
04. var circle = paper.circle(35,50,20);
05. circle.attr('stroke-width', 1);
```

```
06. circle.NewAttributes = { cx: 400, fill: '#088', 'stroke-width': 15, r: 30 };
07. var ellipse = paper.ellipse(35,85,20,10);
08. ellipse.NewAttributes = { cy: 100, ry: 30, rx: 10, cx: 25 };
09. rectangle.attr('fill','#F00').attr('stroke','#444').click( function() { alter(rectangle, 3000); } );
10. circle.attr('fill','#0F0').attr('stroke','#444').click( function() { alter(circle, 1500); } );
11. ellipse.attr('fill','#00F').attr('stroke','#444').click( function() { alter(ellipse, 500); } );
12. function alter(v, t) {
13.     var at = v.attr();
14.     v.animate(v.NewAttributes, t);
15.     v.NewAttributes = at;
16. }
```

Einstieg in CSS

- Über CSS definieren wir extern, intern oder im Element wie Elemente dargestellt werden sollen

```
01. <link rel="stylesheet" href="beispieldatei.css"><!--Externe gesetzte Regeln -->
02. <style> /* Intern gesetzte CSS Regeln  */ </style>
03. <element style="/* Auf dieses Element gesetzte Regeln */" ...>
```

  dazu muss man die entsprechenden Elemente auswählen (Selektor) und Regeln für diese festlegen

- Die Grundlegende Syntax lautet daher immer:

```
01. selektor1, selektor2, ...
02. {
03.     eigenschaft: regel;
04.     /* Weitere Regeln stehen hier! */
05. }
```

- Sog. Responsive Design ist möglich um andere Darstellungen auf Smartphones, Druckern etc. zu bieten
- Um Elemente auszuwählen kann man deren Tagnamen, Struktur, Werte von Attributen sowie die Eigenschaften der Attribute `class` und `id` verwenden
- Einige Eigenschaften vererben an Kindelemente, so dass es z.B. reicht die Schriftart nur einmal zu setzen

Referenzen:
Wikipedia Artikel zu CSS [http://en.wikipedia.org/wiki/Cascading_Style_Sheets]

W3Schools CSS Tutorial [http://www.w3schools.com/css/]
W3C CSS WG mit Statusinfos [http://www.w3.org/Style/CSS/Overview.en.html]
CSS Zen Garten zum guten Design [http://www.csszengarden.com/]
Der CSS Validator [http://jigsaw.w3.org/css-validator/]
SelfHTML CSS Tutorial [http://de.selfhtml.org/css/]

CSS Selektoren

- `*` wählt alle Elemente oder Unterelemente aus - sollte nicht verwendet werden
- `X` wählt alle Elemente mit einem Tag in Form von <X> aus
- `#X` wählt das Elemente mit dem Attribut `id="X"` aus
- `.X` wählt die Elemente mit dem Attribut `class="X"` aus
- `X:Y` wählt die Pseudoeigenschaft Y des Selektors X aus, bspw. `a:hover` oder `a:visited`
- `X~Y` wählt alle Selektoren Y aus die **nach** einem Selektor X folgen
- `X+Y` wählt alle Selektoren Y aus die direkt nach einem Selektor X folgen
- `X  Y` wählt alle Elemente mit dem Selektor Y **unter** dem Selektor X aus
- `X>Y` wählt alle Selektoren Y aus die direkt unter einem Selektor X folgen
- `X[Y]` schränkt den Selektor X auf Elemente ein, die ein Attribut Y besitzen
- `X[Y=Z]` schränkt den Selektor X auf Elemente ein, die ein Attribut Y mit Wert Z besitzen
- `X[Y*=Z]` schränkt den Selektor X auf Elemente ein, die in deren Attribut Y einen Wert Z vorkommt

Referenzen:
W3C über CSS(2) Selectors [http://www.w3.org/TR/CSS2/selector.html]
Tutorial über CSS Selektoren [http://css.maxdesign.com.au/selectutorial/]
W3Schools Selektoren Referenz [http://www.w3schools.com/cssref/css_selectors.asp]
30 sehr wichtige Selektoren [http://net.tutsplus.com/tutorials/html-css-techniques/the-30-css-selectors-you-must-memorize/]
Kleines Tutorial zu Selektoren [http://www.tizag.com/cssT/selector.php]
Ein Demo zum Thema Selektoren [http://html5.florian-rappl.de/selectors.html]

Regeln festlegen

Schema beim Aufbau von CSS-Regeln

- Je nach Element (Tag) kann es spezielle Eigenschaften geben
- Jedes Element besitzt u.a. folgende Eigenschaften
  - `background-color` z.B. `#FF0000`, `red` oder `rgb(255,0,0)`
  - `background-image` z.B. `url('../../beispiel.png')`
  - `background-repeat` mit `no-repeat`, `repeat-x` und `repeat-y`
  - `background-position` mit `x y` z.B. `2px 0` oder `left top`
  - Die oberen vier können in `background` kombiniert werden, z.B.

```
01. #identifier /*Wir legen für das Element mit id=identifier fest*/
02. { /*Hintergrund soll dunkelgrau sein und ein Bild, das sich*/
```

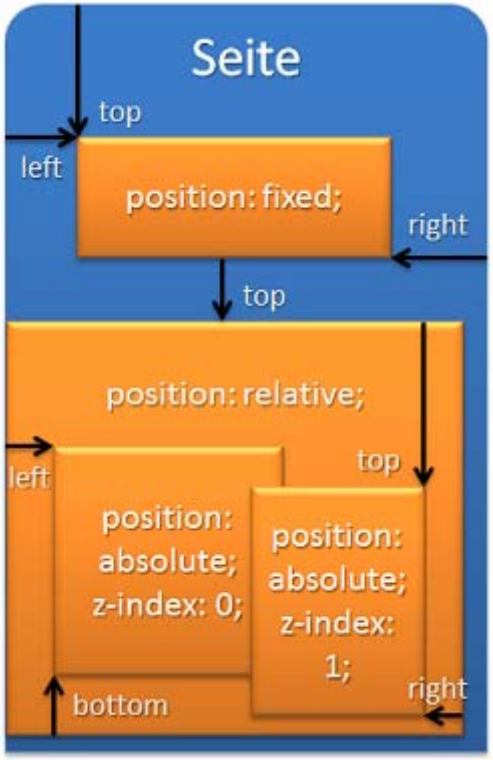
```
03.     background: #777 url('logo.png') no-repeat center center;
04. } /*nicht wiederholt im Zentrum des Elementes besitzen*/
```

- Dieses Prinzip (Kombination) ist nicht auf `background` beschränkt, sondern auch bei anderen Regeln (z.B. `border`) verwendet worden

Referenzen:
CSS Strukturregeln erklärt [http://htmlhelp.com/reference/css/structure.html]
Syntax von CSS Regeln [http://www.w3schools.com/css/css_syntax.asp]
CSS3 Regeln [http://www.javascriptkit.com/dhtmltutors/externalcss3.shtml]
Die @-Regel Referenz [http://reference.sitepoint.com/css/atrulesref]
CSS Background Eigenschaften [http://www.w3schools.com/css/css_background.asp]

Elemente positionieren

- Die Eigenschaft `position` legt die Positionierung von Elementen fest:
  - Bei `position: static` bestimmt der Browser die Position
  - *Bei `position: fixed` bestimmt man die Position relativ im Fenster
  - *Bei `position: relative` ändert man die Stelle relativ zu der vom Browser festgelegten static-Position
  - *Bei `position: absolute` legt man die Position relativ zum ersten Oberelement mit `position: relative` fest
- Die Reihenfolge bei überlappenden Elementen bestimmt `z-index`
- Bei Elementen mit * wird über die Eigenschaften `left` oder `right` und `top` oder `bottom` die X und Y Koordinate festgelegt
- Elemente mit fester Größe (`width`, `height`) bestimmen über die Eigenschaft `overflow` was mit zu viel Inhalt geschieht
- Mögliche Werte sind `hidden`, `scroll` und `visible` - Standard ist `auto`

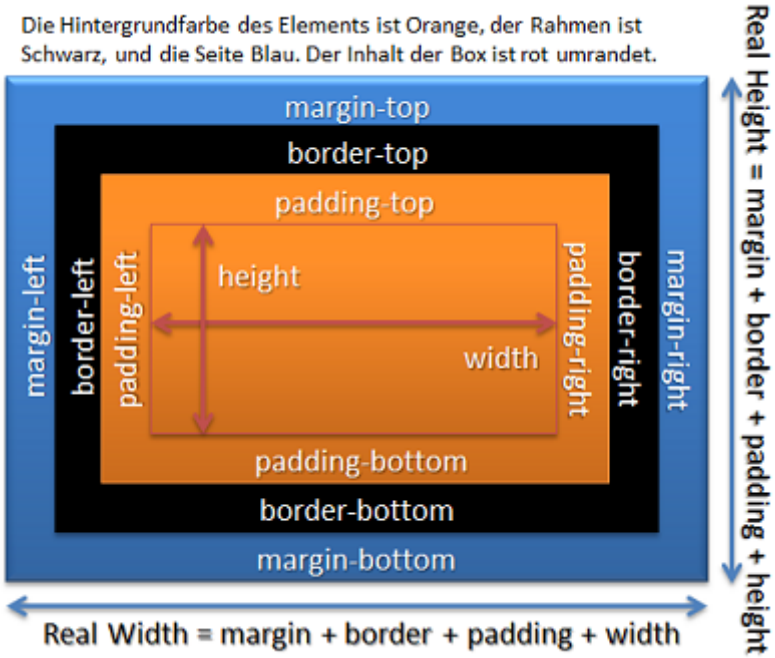


Verdeutlichung der einzelnen Positionierungseigenschaften

Referenzen:
Positionieren von Elementen [http://www.w3schools.com/css/css_positioning.asp]
Dimensionen bei W3Schools [http://www.w3schools.com/css/css_dimension.asp]
Elemente ausrichten bei W3School [http://www.w3schools.com/css/css_align.asp]
Positionieren in 10 Schritten [http://www.barelyfitz.com/screencast/html-training/css/positioning/]
Tutorial zum Positionieren [http://www.html.net/tutorials/css/lesson14.php]

Die Größe festlegen

- Das CSS Box-Model bestimmt die Größe von Elementen auf der Seite (Vorsicht bei IE<6[1])
- Wir können mit `min-width` bzw. `max-width` minimale bzw. maximale Größen einstellen



Die reale Größe eines Elements wird durch die Kombination mehrere Eigenschaften bestimmt

- `width` bzw. `height` stellen eine feste Größe ein
- Über `margin` kann man Abstand zu anderen Elementen einstellen – hier gibt es vier Wege:

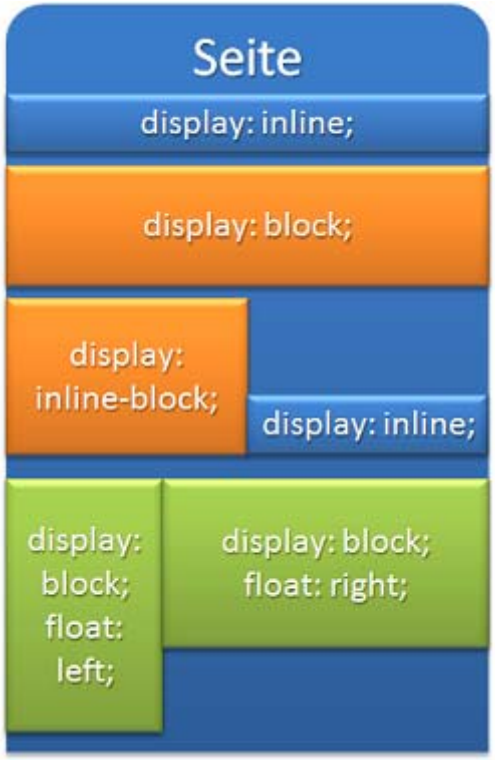
```
01. margin: X; /*Links, Rechts, Oben, Unten Abstand X z.B. 5px*/
02. margin: X Y; /*Oben, Unten Abstand X und Links, Rechts Abstand Y*/
03. margin: O R U L; /*Oben, Rechts, Unten, Links einzeln*/
04. margin-top: 17px; /*Oder direkt*/
```

- `border` setzt den Rahmen (Breite, Stil, Farbe)
- `padding` setzt den inneren Abstand (Breite)

Referenzen:
W3School über das Box-Model [http://www.w3schools.com/css/css_boxmodel.asp]
W3C Artikel zum Box-Model [http://www.w3.org/TR/CSS2/box.html]
Interaktive Demonstration [http://redmelon.net/tstme/box_model/]
Referenz des Box-Models [http://reference.sitepoint.com/css/boxmodel]
[1] Der IE Box-Model Fehler [http://en.wikipedia.org/wiki/Internet_Explorer_box_model_bug]

Anzeigemodus setzen

- Eine Eigenschaft die alle Elemente besitzen ist `display` - bestimmt maßgeblich die Darstellung
- Am häufigsten verwenden wir
  - `block` (Standard bei z.B. , <h1>, <p>)
  - `inline-block` (Standard bei z.B. <img>)
  - `inline` (Standard bei z.B. , <a>)
  - `list-item` (Standard bei z.B.)
  - none (Standard bei z.B. <head>)
- Mit der `float`-Eigenschaft können andere Elemente einen Block umfließen, wobei
  - Bei `float: left` soll das Element so weit links wie möglich sein
  - Mit `float: right` wird das Element rechts fließend ausgerichtet
  - Und bei `clear: both` soll das Element um kein Element herumfließen (im Gegensatz zu z.B. `clear: left`)



Durch die display-Eigenschaft wird der Zeichenmodus bestimmt; mit none gekennzeichnete Elemente werden nicht gezeichnet

Referenzen:

CSS Float bei W3Schools [http://www.w3schools.com/css/css_float.asp]
W3Schools Display Eigenschaft [http://www.w3schools.com/css/css_display_visibility.asp]
Block vs Inline Tutorial [http://www.webdesignfromscratch.com/html-css/css-block-and-inline/]
CSS Display Tutorial [http://www.tizag.com/cssT/display.php]
Layoutmöglichkeiten mit CSS im Detail [http://webdesign.about.com/od/beginningcss/p/aacss9layout.htm]

Beispiel (11)

Der HTML Code:

```
01. <div style="position: absolute; z-index: 3; background: #F00; border: 5px solid rgb(0,255,0); width: 200px; height: 100px; left:
    20px; top: 400px; text-align: center; margin: 10px; padding: 10px;" id="abs-TestBox1">Hallo!</div>
02. <div style="position: absolute; z-index: 2; background: #00F; border: 1px solid rgb(0,0,0); width: 200px; height: 100px; left:
    20px; top: 400px; text-align: center; margin: 0; padding: 0; color: white;" id="abs-TestBox2">Hallo!</div>
03. <div style="position: relative; top: 400px; background: #F0F; border: 1px solid rgb(0,0,0); width: 50px; height: 50px; color:
    white;" id="abs-TestBox3"></div>
```

Der jQuery Code:

```
01. $('#abs-TestBox1').click(function() { $(this).css({margin: 0, border: 0, 'z-index': 1}); });
02. $('#abs-TestBox2').click(function() { $(this).css({left: 30, top: 410}); });
03. $('#abs-TestBox3').click(function() { $(this).animate({marginLeft: 700}, 2000); });
```

CSS im Detail

- Mit Hilfe von CSS legt man Designregeln fest (Write once use anywhere)
- Durch CSS läßt sich Design sehr schnell und einfach verändern
- Ein weiterer Vorteil von CSS liegt darin, dass jede Menge Traffic gespart wird (kürzere Ladezeit)
- Neben den Verwenden von Tagnamen, ID, Klasse, Attributen und Pseudo-Selektoren kann man Selektoren auch Gruppieren: Durch ein Komma (,) getrennte Selektoren werden kombiniert, z.B. geht

  ```
  01. h1 { /* Regeln */ }
  02. h2 { /* Die gleichen Regeln wie h1 */ /* Eigene Regeln */ }
  03. /* Viel besser wäre */
  04. h1, h2 { /* Gemeinsame Regeln */ }
  05. h2 { /* Eigene Regeln */}
  ```

- Einige Pseudoklassen (sehr nützlich um optische Differenzierung bei Interaktionen zu ermöglichen!):
  - Vorher neben `:focus` v.a. für Links bekannt (`:visited`, `:active`, `:hover`, `:link`)
  - Mit CSS2 `:first-child`, `:last-child`, `:before`, `:after`, `:lang(id)`, `:first-letter`, `:first-line`)
  - In CSS3 `:nth-child(N)`, `:nth-last-child(N)`, `:nth-of-type(N)`, `:enabled`, `:disabled`, `:empty`, ...

Referenzen:
CSS3 Pseudoklassen [http://reference.sitepoint.com/css/css3psuedoclasses]
Wie man die CSS3 Pseudoklassen nutzt [http://coding.smashingmagazine.com/2011/03/30/how-to-use-css3-pseudo-classes/]

W3Schools über Pseudoklassen [http://www.w3schools.com/css/css_pseudo_classes.asp]
Den CSS Selektoren kennenlernen [http://css-tricks.com/5762-pseudo-class-selectors/]
Kompletter Guide zu CSS Selektoren [http://www.suburban-glory.com/blog?page=125]

Möglichkeiten

- Nochmals zum Einbinden von CSS - die drei Möglichkeiten beleuchtet:
  - Als `style`-Attribut im entsprechenden Element: Nur verwenden wenn sich die Regeln **nie** wiederholen
  - Im `<style>`-Tag: Wenn die Regeln **nur** auf dieser Seite vorkommen (und es nicht zu viele sind)
  - Im `<link>`-Tag: Wenn die Regel auf **mehreren** Seiten vorkommt (oder um schneller zu laden)
- Was hat man über CSS noch für Möglichkeiten:
  - In CSS kann man auch den Mauszeiger (`cursor` Eigenschaft) einstellen, z.B. ist `default` der Standardzeiger und `pointer` die Hand:

```
01. #beispiel { cursor: default; }
02. #beispiel:hover { cursor: pointer ; }
```

  - Über `@import` Regel können (weitere) externe CSS Dateien in CSS eingelesen werden
  - Konditionen gehen auch über @-Anweisungen (z.B. spezielle Darstellung bei kleineren Auflösungen)
  - Mit CSS3 können wir auch Berechnungen durchführen z.B. durch `calc(75% - 100px)`
  - Bei Berechnungen sollte man allerdings aufpassen - dies war schon früher möglich führte jedoch häufig zu rekursiven Aufrufen!

Referenzen:
W3Schools über die Möglichkeiten [http://www.w3schools.com/css/css_howto.asp]
Möglichkeiten zum Einbinden [http://www.webstartcenter.com/howto/css_inc.php]
Einbinden von externen CSS Dateien [http://stackoverflow.com/questions/147500/is-it-possible-to-include-one-css-file-into-another]
Die Cursor Anweisung [http://www.w3schools.com/cssref/pr_class_cursor.asp]
Cursors Artikel von Quirksmode [http://www.quirksmode.org/css/cursor.html]
Älteres Tutorial zu Cursors [http://www.echoecho.com/csscursors.htm]

Schrift und Text

- Schriften mit CSS - Alle `font-`-Anweisungen können in `font` zusammengefasst werden
- `font-family` erwartet einen String (Schriftart) als Argument, Reihenfolge bestimmt Präferenzen
- `font-style` mit (*normal*, *italic*, *oblique*) bestimmt den Anzeigestil
- `font-variant` mit (*normal*, *small-caps*) bestimmt ebenfalls den Darstellungsstil
- `font-weight` mit (*normal*, *bold*, *bolder*, *lighter*, *100-900*) bestimmt die Dicke der Schrift

- `font-size` erwartet rel. Strings (*normal*, *smaller*, *larger*, ...) oder tot. Angaben z.B. *12pt*, *14px* oder *75%* etc.
- `line-height` mit Größenangaben wie z.B. *22px*, *150%* oder *1.5* (relativ)
- `text-decoration` gibt vor ob (*none*, *underline*, *overline*, *line-through*, *blink*) gilt
- `text-transform` kann weitere Darstellungsstile wie z.B. (*none*, *capitalize*, *uppercase*, *lowercase*) einstellen
- `text-align` definiert die Textausrichtung (*left*, *right*, *center*, *justify*)
- `white-space` stellt ein wie leerzeichen im Code interpretiert werden (*normal*, *pre*)
- `color` stellt die Schriftfarbe ein in z.B. hex *#112233* bzw. *#123* oder *rgb(r,g,b)*

Referenzen:
W3Schools über Schrift-Styling [http://www.w3schools.com/css/css_font.asp]
W3Schools über Text-Styling [http://www.w3schools.com/css/css_text.asp]
Schriftarten und CSS [http://www.somacon.com/p334.php]
Text mit CSS [http://www.echoecho.com/csstext.htm]
Tutorial über CSS (Textsatz) [http://www.htmlite.com/CSS004.php]
Schriften mit CSS [http://www.tizag.com/cssT/font.php]

Mehr Schriftarten!

- Mit CSS3 haben wir die Möglichkeit eigene Schriften zu definieren und zu verwenden
- Wir können eigens auf unserem Server gespeicherte Schriften verwenden oder extern gespeicherte
- Die Syntax hierfür definiert in CSS eine neue Schriftart, z.B.

```
01. @font-face {
02.     font-family: 'Gentium'; /* Wird der neue Name! */
03.     src: local('Gentium'), url('../fonts/Gentium.ttf'); /* Bevorzugt nicht downloaden */
04.     font-style: italic; /* Optional: Charakterisiert Schriftart z.B. Verwendung nur bei italic*/ }
05. #Verwendung { font: italic bold 14pt 'Gentium'; }
```

- Formate? true-type-font (**ttf**) geht mittlerweile überall - bevorzugt ist **WOFF** (Web Open Font Format)
- Über Google Webfonts[1] kann man an Schriften kommen, die direkt über den Google Server abrufbar sind
- Eine sehr stark Browser-unabhängige Schreibweise für die `@font-face` Regel is

```
01. @font-face { font-family: 'Whatever'; /* Erstellen neue Schriftart Whatever */
02.     src: url('myfont.eot'); /* IE9 Compatibility Modus - Liefert Support für IE ab Version 4*/
03.     src: url('myfont.eot?iefix') format('eot') /*IE6-IE8*/, url('myfont.woff') format('woff') /*Modern*/,
```

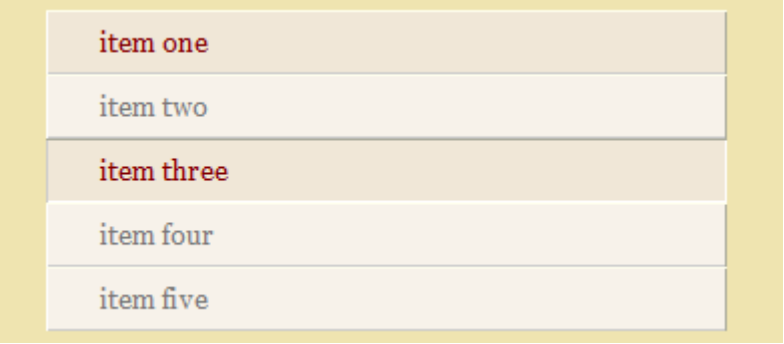
```
04.          url('myfont.ttf') format('truetype'), /*Mobile*/ url('myfont.svg') format('svg'); /*Fallback*/ }
```

Referenzen:
Sammlung von OpenSource Schriften [http://www.theleagueofmoveabletype.com]
Sehr gute Seiten mit kostenlosen Schriften [http://www.fontsquirell.com/fontface/]
[1]Google Webfonts [http://www.google.com/webfonts]
W3Schools über Fonts [http://www.w3schools.com/cssref/pr_font_font.asp]
Tool zur Suche nach Fallbackschriften [http://www.ffffallback.com]

Listen richtig stylen

- Listen eignen sich sehr gut für Menüs, da man hier sehr schönes Styling vornehmen kann und die Abwärtskompatibilität perfekt ist
- `list-style-type` kann durch Angabe von *disc*, *circle*, *square*, *decimal*, *lower-roman*, *upper-roman*, *lower-alpha*, *upper-alpha*, *none* gesetzt werden
- Über `list-style-image` läßt sich ein Bild für die Auflistung einstellen, d.h. *url(...)*
- `list-style-position` definiert wo sich der Punkt befinden soll, d.h. *inside* oder *outside*
- Alle Angaben können über `list-style` zusammengefasst werden

Das Aussehen der Liste ohne CSS auf die Listenelemente



Durch CSS hat sich das Aussehen der Liste komplett verändert

```
01. ul#navlist { text-align: left; list-style: none; padding: 0; margin: 0 auto; width: 70%; }
02. ul#navlist li { display: block; margin: 0; padding: 0; }
```

```
03. ul#navlist li a { display: block; width: 100%; padding: 0.5em 0 0.5em 2em; border-width: 1px; border-color: #ffe #aaab9c #ccc
    #fff; border-style: solid; color: #777; text-decoration: none; background: #f7f2ea; }
04. ul#navlist li#active a { background: #f0e7d7; color: #800000; }
```

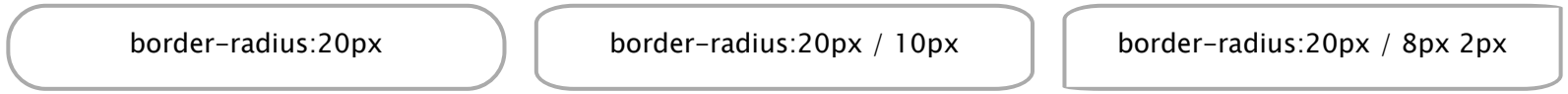
Referenzen:
W3Schools Artikel [http://www.w3schools.com/css/css_list.asp]
Listen zähmen [http://www.alistapart.com/articles/taminglists/]
Auswahl von Listen [http://css.maxdesign.com.au/listamatic/]
List-Style Wizard [http://www.somacon.com/p357.php]
8 tolle Listenbeispiele [http://www.marcofolio.net/css/8_different_ways_to_beautifully_style_your_lists.html]
Mehr über ungeordnete Listen [http://www.mediacollege.com/internet/css/ul-list.html]
Mehr über geordnete Listen [http://codeasily.com/css/style-ordered-list]

Der richtige Rahmen

- Bereits erwähnt wurde die `border` Eigenschaft - hierzu ein Beispiel:

```
01. #someBox {
02.     border: 1px solid #CCC; /*Oben, rechts, unten, links 1px durchgezogen grau*/
03.     border-left: 2px dashed #F00; /*Links 2px gestrichelt rot*/
04.     border-bottom: 1em dotted green; /*Unten 1em (1 M in aktueller Schriftgröße) gepunktet Grün*/ }
```

- Mit CSS3 wurden Rahmen deutlich flexibler - wir können nun über `border-radius` **runde Ecken** erzeugen!
- Wir setzen `border-radius: lo ro ru lu` von links oben nach links unten
- Der Teufel liegt im Detail: Man kann auch mit zwei Angaben arbeiten - sozusagen elliptische Ecken!
- z.B. über `border-radius: lo1 ro1 ru1 lu1 / lo2 ro2 ru2 lu2` - ist immer äquivalent ist zu:
- den einzelnen Angaben z.B. `border-top-left-radius: lo1 lo2`



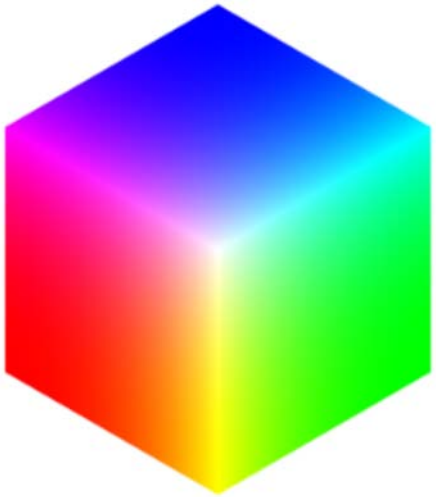
Referenzen:
Robert Nyman zum Border-Radius [http://robertnyman.com/css3/border-radius/border-radius.html]
W3Schools über border-style [http://www.w3schools.com/cssref/pr_border-style.asp]
Runde Ecken am Anfang (Apple v Mozilla) [http://www.css3.info/border-radius-apple-vs-mozilla/]

W3Schools zu den runden Ecken [http://www.w3schools.com/cssref/css3_pr_border-radius.asp]
Mit border-radius rumspielen [http://www.the-art-of-web.com/css/border-radius/]

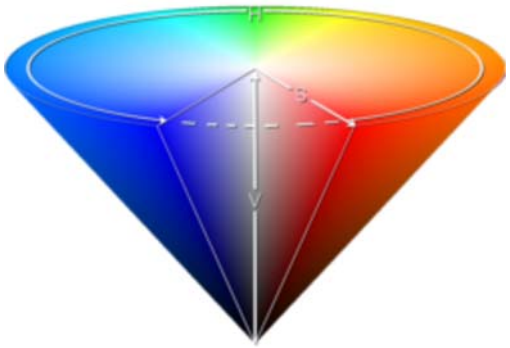
Farben in CSS3

- In CSS gibt es mehrere Wege Farben zu definieren, z.B. über den Namen `color: red`
- Es gibt auch noch die hexadezimale Darstellung, z.B. `background: #FF2266`
- Diese Darstellung wird immer im RGB-Modell durchgeführt, und kann auch Dezimal geschrieben werden: `background: rgb(255, 34, 102)`
- In CSS3 wurden transparente Farben eingeführt - das sog. RGBA-Modell. Um dies nutzen zu können schreibt man z.B. `color: rgba(255, 32, 102, 0.5)`, wobei hier Alpha auf 50% steht
- Ein Alpha-Wert von 100% (1.0) bedeutet volle Deckung (keine Transparenz)
- Daneben wurde in CSS3 das HSL (Hue Saturation Light, d.h. Farbwinkel Sättigung Licht) eingeführt

```
01. #hslexample /*Sättigung von 70%*/
02. { border-top: 2px solid hsl(120, 70%, 50%); }
03. /*Farbwinkel von 120° entspricht Blau*/
```



Ausschnitte aus dem RGB Modell (Rot, Grün, Blau)



Ausschnitte aus dem HSV Modell (Farbwinkel, Sättigung, Licht)

Referenzen:
CSS3 Farb-Erweiterungen [http://www.w3.org/TR/css3-color/]
HSL in CSS3 [http://www.css3.info/preview/hsl/]
Wikipedia Artikel zu HSL (und HSV) [http://en.wikipedia.org/wiki/HSL_and_HSV]
Seite zum Testen von Farben [http://html5.florian-rappl.de/colors.html]
RGBA bei CSSinfo [http://www.css3.info/preview/rgba/]
Namen von vordefinierten Farben [http://xilize.sourceforge.net/Reference/colorref.html]

Beispiel (12)

```
01. @font-face {
02.     font-family: 'Ubuntu'; /* Verwenden die populäre Schrift Ubutunu die wir von Google kriegen */
03.     src: local('Ubuntu'), url('.../_xyN3apAT_yRRDeqB3sPRg.woff') format('woff'); /* Wuff Wuff */
04. }
05. #beispiel12 {
06.     list-style-type: none; /* Schalten den Listen-Typ ab - keine Aufzählungszeichen */
07.     font: 16pt 'Ubuntu'; /* Verwenden unsere eigene Schrift */
08.     background: hsl(25, 90%, 60%); /* Hintergrund über HSL definieren */
09.     border-radius: 30px 30px 15px 15px / 15px; /* Runde Ecken versteht sich, oben elliptisch */
10. }
11. #beispiel12 > li:nth-child(odd) { background: rgba(140, 200, 92, 0.2); } /* Ungerade li */
12. #beispiel12 > li:nth-child(even) { background: rgba(140, 200, 92, 1); } /* Gerade li */
13. #beispiel12 > li {
14.     color: white; vertical-align: middle; /* Textfarbe und Ausrichtung */
15.     display: inline-block; /* Anzeige in einer Zeile */
```

```
16.     border: 1px solid #CCC; /* Rahmen bestimmen */
17.     margin: 0 25px; /* Abstand 0 nach oben und unten, 25 pixel links und rechts */
18.     width: 150px; height: 50px; /* Setzen der Größe */
19. }
20. #beispiel12 > li:empty { border: 1px dashed #F00; } /* Regel für leere li */
```

Farbverläufe

- Im Moment ist die Spezifikation für Farbverläufe noch in permanenter Bewegung
- Es läuft aber wohl ganz klar auf diese Syntax raus:

```
01. .linear { background-image: linear-gradient(top, #00f, #fff); }
02. .linearStop { background-image: linear-gradient(left top, #00f, #fff 80%); }
03. .linearMultiple { background-image: linear-gradient(left, #F00, #0F0 50%, #00F); }
04. .radial { background-image: radial-gradient(40% 40%, farthest-side, #ffffa2, #00f); }
```

- Die Gradienten funktionieren leider (wie viele andere CSS3) Features im Moment nur über Browser-Spezifische Präfixe - nehmen wir als Beispiel `linear-gradient(...)`
- Daraus wird in Opera `-o-linear-gradient(...)` und in Firefox `-moz-linear-gradient`
- Microsoft hat als Präfix `-ms-`, Apple und Google verwenden `-webkit-`
- Wir werden uns später noch mit den Präfixen beschäftigen
- Eine Möglichkeit die Präfixe zu umgehen ist JavaScript[1]

Referenzen:
Robert Nyman zu Gradienten [http://robertnyman.com/css3/gradients/gradients.html]
[1]Prefix-Free JavaScript [http://leaverou.github.com/prefixfree/]
Gradient-Generator [http://gradients.glrzad.com/]
Noch ein (mächtigerer) Generator [http://www.westciv.com/tools/gradients/]
Webkit Tutorial zu Gradienten [http://www.webkit.org/blog/1424/css3-gradients/]
Artikel über CSS3 Farbverläufe [http://www.zenelements.com/blog/css3-gradients/]

Licht und Schatten

Die erste Box mit einem Schatten nach rechts unten (Dunkelgrau)

Die zweite Box mit einem Schatten nach links unten (Rot)

Die dritte Box mit einem Textschatten nach rechts unten (Grau)

Die vierte Box mit einem Textschatten nach links unten (Grün)

- Mit CSS3 sind Schatten für Texte und Boxen möglich
- Die Syntax für Boxen lautet

```
01. #eins { box-shadow: 5px 5px 3px #666; }
02. #zwei { box-shadow: -5px 5px 10px #F00; }
```

- Bei Text ist die Syntax sehr ähnlich z.B.

```
01. #drei { text-shadow: 3px 3px 5px #999; }
02. #vier { text-shadow: -3px 3px 2px #0F0; }
```

- Die Syntax ist also immer so aufgebaut:
  - Versatz horizontal (pos. rechts, neg. links)

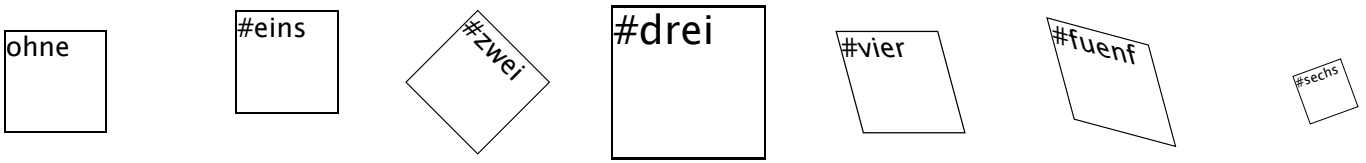
- Versatz vertikal (pos. unten, neg. oben)
- Verschwommenheit (0 entspricht klar)
- Farbe des Schattens

Referenzen:
Schatten auf Boxen [http://robertnyman.com/css3/drop-shadow/drop-shadow.html]
Schatten auf Text [http://robertnyman.com/css3/text-shadow/text-shadow.html]
Boxschatten mit CSS3 erklärt [http://www.css3.info/preview/box-shadow/]
Textschatten mit CSS3 erklärt [http://www.css3.info/preview/text-shadow/]
Tipps zum UI Styling mit Schatten [http://dev.opera.com/articles/view/beautiful-ui-styling-with-css3-text-shadow-box-shadow-and-border-radius/]

Transformationen

- Neu in CSS3 sind auch Koordinatentransformationen - hier haben wir mehrere Möglichkeiten:
  - Translation mit `translate(dx,dy)` zum Verschieben (Standard: 0, 0)
  - Rotation mit `rotate(alpha)` zum Drehen (Standard: 0)
  - Skalieren mit `scale(zoomfaktor)` zum Verändern der Größe (Standard: 1)
  - Verzerren mit `skew(phi,psi)` zur Schrägstellung (Standard: 0, 0)

```
01. #eins { transform: translate(10px, -10px); }
02. #zwei { transform: rotate(45deg); }
03. #drei { transform: scale(1.5); }
04. #vier { transform: skewX(15deg); }
05. #fuenf { transform: skew(15deg, 15deg); }
06. #sechs { transform: scale(0.5) rotate(-20deg) translate(0, 10px);  }
```



- Daneben sind noch 3D-Transformationen möglich, hier ist die Unterstützung allerdings noch zu gering

Referenzen:
Artikel über die Möglichkeiten mit transform [http://www.zenelements.com/blog/css3-transform/]
W3Schools über Transform [http://www.w3schools.com/cssref/css3_pr_transform.asp]
Die W3C Spezifikation [http://www.w3.org/TR/css3-2d-transforms/]
Artikel über Transitions, Animationen und Transforms [http://css3.bradshawenterprises.com/]
Transformationen auf Robert Nymans Seite [http://robertnyman.com/css3/css-transforms/css-transforms.html]

Effekte mit CSS3

- Vorher waren Änderungen am CSS Stil z.B. durch `:hover` beim Überfahren, immer direkt
- Mit CSS3 haben wir die Möglichkeit Änderungen an Eigenschaften durch Pseudoklassen zu animieren
- Dies wird als **Transition** bzw. Effekt bezeichnet (nicht als Animation - dazu gleich mehr)
- Die Syntax benutzt dazu folgende Elemente (kann wiederholt durch `,` angegeben werden):
  - Welche Eigenschaften sind von der Transition betroffen? (`all` meint alle)
  - Wie lange soll der Effekt gehen? z.B. `1s` oder ab wann und wie lang z.B. `0.5s 1s`
  - Welcher Effekt soll genau benutzt werden? z.B. ease (Standard) oder `linear`

```
01. .beispiele:hover { background: #F00; width: 100px; } .beispiele { background: #0F0; width: 50px; }
02. #eins { transition: background-color 1s linear; }
03. #zwei { transition: background-color 1s ease; }
04. #drei { transition: background-color 1s ease-in; }
05. #vier { transition: background-color 1s ease-out; }
06. #fuenf { transition: background-color 1s ease-in-out, width 0.5s 1s linear; }
```

Referenzen:
Beispiel zu einfachen Transitions [http://robertnyman.com/css3/css-transitions/css-transitions-simple.html]
Beispiel zu Transitions mit Transforms [http://robertnyman.com/css3/css-transitions/css-transitions-rotation.html]
Beispiel: MacOS-X Stacks [http://robertnyman.com/css3/css-transitions/css-transitions-mac-os-x-stacks.html]
Artikel über Transitions, Animations und Transforms [http://css3.bradshawenterprises.com/]
Tutorial über Transitions [http://net.tutsplus.com/tutorials/html-css-techniques/css-fundametals-css-3-transitions/]
W3Schools über CSS3 Transitions [http://www.w3schools.com/css3/css3_transitions.asp]

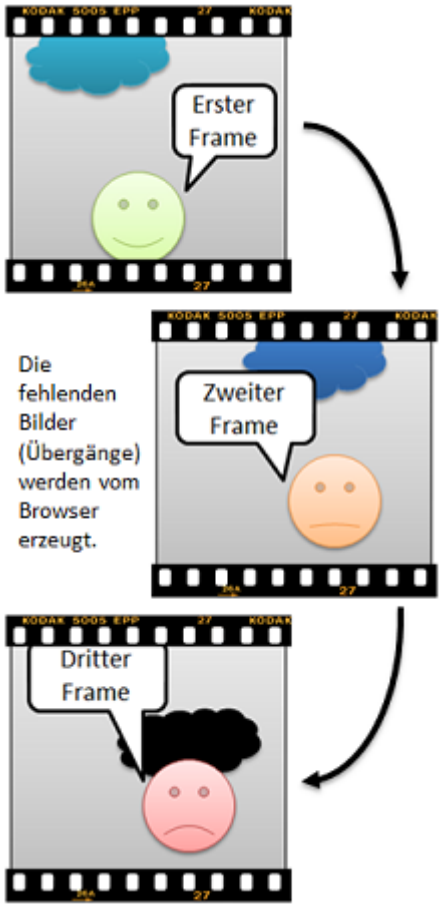
Animationen ohne JS

- Effekte wie eben können wir natürlich auch ohne CSS3 in JavaScript erledigen (per CSS ist schneller!)
- Wie sieht es aber nun aus wenn wir nicht nur Übergänge handhaben wollen sondern echte Animationen?
- Selbst hier können wir auf JavaScript verzichten - CSS3 Animationen!

```
01. @-webkit-keyframes beispiel { /* Definition der Keyframes */
02.     0% { background-color: #F00; } /* Erster Frame am Start */
03.     20% { background-color: #0F0; } /* Frame bei 20% der Zeit */
04.     60% { background-color: #00F; } /* Dritter Frame */
05.     100% { background-color: #FFF; } /* Abschlussframe */
```

```
06. }
07. #someDiv {
08.     -webkit-animation-name: beispiel; /* Verweis auf keyframes */
09.     -webkit-animation-duration: 10s; /* Analog Transitions */
10.     -webkit-animation-iteration-count: infinite; /* auch 1, ... */
11.     -webkit-animation-timing-function: linear; /* Analog Trans. */
12. }
```

- Animationen gehen bislang nur auf Safari, Chrome, Firefox (`-moz-`)



Die Schlüsselbilder legen die Fixpunkte fest

Referenzen:
Artikel über Animationen, Transitions und Transforms [http://css3.bradshawenterprises.com/]

Einfache CSS3 Animation [http://robertnyman.com/css3/css-animations/css-animations-simple.html]
Schwierigeres Beispiel [http://robertnyman.com/css3/css-animations/css-animations-percentage.html]
Film in CSS3 Transitions [http://stuffandnonsense.co.uk/content/demo/madmanimation/]
Tool zur Erstellung von CSS3 Animationen [http://animatable.com]

Neue Layout-Möglichkeiten

- Mit CSS3 geht auch die Ära der div-Layouts dem Ende zu - wir haben nun besseres
- Natürlich kehrt man nicht zur guten alten Tabelle zurück, sondern zu spezialisierten Layout-Mustern
- Bisher am weitesten verbreitet sind Mehrspalten (siehe Inhaltsverzeichnis), z.B.

```
#zBeinDIV { /* Wir legen nur eins von beiden fest: Entweder */
    column-count: 3; /* Feste Anzahl oder Feste Breite */ column-width: 100px;
    column-gap: 10px; /* Abstand zwischen den Spalten */
    column-rule-width: 1px; /* Breite der mittleren Trennlinie */
    column-rule-color: #CCC; /* Farbe der Trennlinie - leicht grau */
    column-rule-style: dashed; /* Analog zu den Styles bei border */ }
#zBeinDIV h1 { column-span: 3; } /* Geht über 3 Spalten hinweg */
```

- Aber damit nicht genug! Es werden auch **flexible Boxen** mit `display: flexbox` eingeführt
- Daneben soll noch ein Grid-Layout kommen (soz. Tabelle für Layouts) wie in Windows 8 Metro
- Bei ersterem ist die Spezifikation zur Zeit noch unter ständiger Arbeit
- Bei letzterem ist bis jetzt noch unklar ob die anderen Browserhersteller auf Microsofts Entwurf eingehen

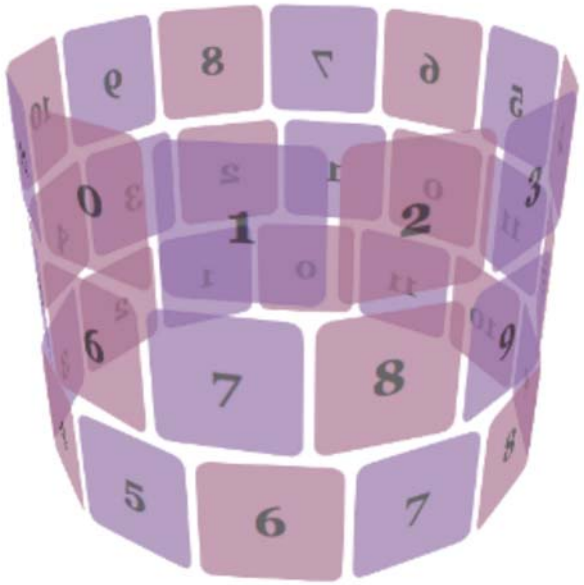
Referenzen:
W3C Dokument zu Mehrspalten [http://www.w3.org/TR/css3-multicol/]
W3C Aktuelles zur Flexbox [http://dev.w3.org/csswg/css3-flexbox/]
Artikel über CSS3 Mehrspalten [http://www.zenelements.com/blog/css3-multiple-columns/]
CSS3 Multi Column Tutorial [http://benjamin-weigl.de/journal/tutorials/css3-multi-columns.html]
Vorstellung der flexiblen Boxen [http://www.css3.info/introducing-the-flexible-box-layout-module/]
Sanfte Einführung in Flex-Box [http://www.the-haystack.com/2010/01/23/css3-flexbox-part-1/]

Beispiel (13)

```
#beispiel13 { /* Wählen Farbverlauf als Hintergrund */
    background-image: linear-gradient(top, #F00, #FF0 25%, #0F0 50%, #0FF 75%, #00F);
    transition: all 1s ease; /* Wollen Übergänge beim Ändern JEDER Eigenschaft */
```

```
    width: 200px; height: 30px;    padding: 20px; /* Festlegen der Größe */
    text-shadow: 2px 2px 2px #000; box-shadow: 5px -5px 2px #F0F; /* Schatteneffekte */
    color: white; font-size: 20pt; text-align: center; /* Texteinstellungen */
}
#beispiel13:hover { /* Veränderung beim Drüberfahren */
    transform: rotate(30deg) scale(1.2); /* Skalieren und Rotieren! */
}
#beispiel13:empty { /* Veränderungen wenn es eine leere Box ist */
    transform: skew(45deg, 0) scale(0.5); /* Verzerren und Skalieren */
    background: #CCC; /* Grauer Hintergrund */
}
```

3D Effekte mit CSS3



Boxen können nun in 3D durch CSS3 angeordnet und animiert werden

- Mit CSS3 kann man auch 3D Transformationen einer Webseite hinzufügen
- Die Vorteile liegen auf der Hand:
    - Der Browser kann mit Objekten viel besser Umgehen als mit Zeichnungen auf einem Canvas

    - Somit erhält man Vorteile durch mögliche Hardwarebeschleunigung
    - Text und weitere Elemente (z.B. auch Videos etc.) kann man direkt in HTML leichter einbinden als in einem Canvas
    - HTML und CSS benötigen keinen Code und laufen somit selbst auf Browsern mit deaktiviertem JavaScript
- Prinzipiell wird die bekannte Syntax zu Transformationen ergänzt, z.B.

```
... { transform: perspective(5); } /*Stellt Perspektive ein*/
... { transform: scaleZ(2); } /*Skaliert die z-Koordinaten*/
... { transform: rotateZ(10deg); } /*Rotiert um z-Achse*/
... { transform: skewZ(15px); } /*Verzerrt z-Richtung*/
```

Referenzen:
20 tolle Beispiele [http://www.netmagazine.com/features/20-stunning-examples-css-3d-transforms]
3D Transformationen [http://css3.bradshawenterprises.com/]
Flip-Card Beispiel [http://css3playground.com/flip-card.php]
W3C über 3D Transformationen [http://www.w3.org/TR/css3-3d-transforms/#transform-property]

CSS3 ohne Präfixe

- Bis jetzt haben wir die Präfixhölle unter den Tisch gekehrt (`-ms-`, `-o-`, `-webkit-`, `-moz`)
- Der Grund liegt darin, dass es sich bei den entsprechenden Eigenschaften noch nicht um endgültig abgesegnete Spezifikationen handelt
- Neben der Lösung, dass man die zusätzlich notwendigen Zeilen selbst schreibt (bzw. kopiert, einfügt und dann das Präfix ändert) gibt es noch elegantere Lösungen
- Eine Möglichkeit besteht darin, Seiten wie prefixmycss[1] zu verwenden - hier fügt man seinen Code ein und erhält den entsprechenden Code zurück
- Der Vorteil an dieser Methode liegt darin, dass eben kein JavaScript aktiviert sein muss damit das CSS richtig funktioniert
- Wenn JavaScript aktiv ist, stellt die Bibliothek prefixfree[2] eine ausgezeichnete Lösung dar
- Der Vorteil ist, dass man die CSS Datei nicht unnötig aufbläht und nur die Präfixes angewandt werden, die notwendig sind
- Ein sauberer CSS Code der auf Präfixe verzichtet ist sicherlich die Zukunft
- Ein Beispiel für prefixfree stellt dieses Skript dar (siehe Quellcode)

Referenzen:

[1] CSS Code Modifizierer [http://prefixmycss.com/]

[2] JavaScript Lösung Prefixfree [http://leaverou.github.com/prefixfree/]
Warum CSS Präfixe? [http://www.css3.info/why-and-when-browsers-prefix-css3-features/]
Entscheiden ob ein Präfix gebraucht wird [http://stackoverflow.com/questions/3748512/dynamic-css3-prefix-user-agent-detection]

CSS3 für ältere Browser

- Wir haben durch die Präfixe schon Probleme mit euen Browsern: können CSS3-Möglichkeiten überhaupt in ältere Browser eingebaut werden?
- Im Prinzip sollte man Webseiten je nach Projekt so aufbauen, dass diese auch ohne CSS3 funktioniert
- Ein möglicher Weg besteht darin, die notwendigen Features zu erkennen und einen entsprechenden Fallback einzubauen
- Ein anderer Weg besteht darin, einige CSS Eigenschaften mehrfach zu benennen
- Beispiel: Wir verwenden einen CSS3 Gradienten um einen Farbverlauf im Hintergrund zu haben, also

```
01. body { /*Selektor um das body Element auszuwählen*/
02.     background-image: url(gradient.png) repeat-x; /*Fallback für ältere Browser*/
03.     background-image: linear-gradient(/*Farbverlauf*/); /*Für moderne Browser*/ }
```

  weshalb die erste Eigenschaft durch die zweite auf modernen Browsern überschrieben wird
- Eine weitere Möglichkeit bietet CSS PIE für den IE - hier werden sog. `filter` ausgenutzt
- Eine Beispiel für einen Filter stellt z.B. `rgba()` auf älteren IEs dar:

```
01. filter: progid:DXImageTransform.Microsoft.gradient( startColorstr=#7BFFFFFF, endColorstr=#7BFFFFFF );
```

Referenzen:
Artikel von Smashing Magazine [http://coding.smashingmagazine.com/2011/05/03/using-css3-older-browsers-and-common-considerations/]
CSS3 Attribute in IE6-8 [http://selectivizr.com/]
Feature Detection und viele Möglichkeiten [http://www.modernizr.com/]
Farbverlauf, Runde Ecken und Schatten für den IE [http://css3pie.com/]

Neue Tags für ältere IE

- Sehr alte Browser besitzen natürlich keine Multimedia Fähigkeiten und nur limitierte CSS(3) Fähigkeiten
- Ein Problem erhalten wir jedoch, wenn die neuen (semantischen) HTML5-Tags auch nicht funktionieren
- Zwar sollte jeder Browser dann ein aus unbekannten Tags machen - allerdings keine sehr alten IE
- Hier hilft folgender Workaround im <head>-Bereich einer Seite:

```
01. <script type="text/javascript">
02. document.createElement('header');
03. document.createElement('footer');
04. document.createElement('article');
05. //etc... (je nachdem was gebraucht wird
```

```
06. </script>
```

- Dieser doch rechte einfache Workaround erstellt einfach Elemente mit den neuen Namen und gibt den IE dadurch an, dass es diese Elemente wirklich geben soll - die Elemente können dann verwendet werden
- Daneben ist manchmal noch folgende Angabe zur korrekten -Emulation notwendig:

```
01. <style type="text/css"> header, footer, article, ... { display: block; }
02. /* Sorgt dafür dass die Elemente ANFANGS wirklich wie divs ist */</style>
```

Referenzen:
Neue Elemente in alten Browsern? [http://pietschsoft.com/post/2010/11/14/HTML5-Day-1-New-tags-work-in-older-browsers-Awesome.aspx]
JavaScript für perfekte Kompatibilität [http://code.google.com/p/html5shim/]
Workaround für alte IEs [http://net.tutsplus.com/tutorials/html-css-techniques/how-to-make-all-browsers-render-html5-mark-up-correctly-even-ie6/]
StackOverflow über neue Tags für alte IE [http://stackoverflow.com/questions/4175050/html5-new-layout-elements-old-browsers]

Canvas in alten Browsern

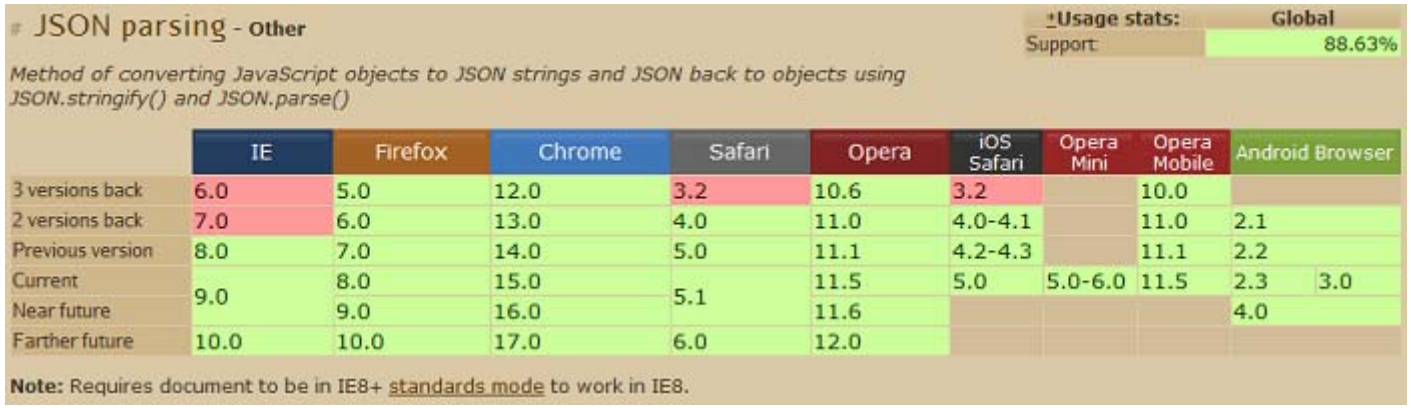
- Fallback für Audio und Video erhalten wir durch Plugins wie Flash oder Warnungen (mit Download-Links)
- Doch was für Möglichkeiten haben wir bei Zeichnungen mit Canvas?
- Hier bieten sich wie `filter`-Eigenschaften für ältere IEs an
- Die Verwendung dieser Möglichkeiten in JavaScript sind jedoch genauso kompliziert wie in CSS
- Daher gibt es eine einfache Lösung in der Verwendung der JavaScript Bibliothek exCanvas
- Diese gibt uns die Möglichkeit einen Canvas-Context in älteren IEs zu verwenden mit Hilfe von

```
01. <!--[if IE]>
02. <script src="excanvas.js"></script>
03. <![endif]-->
```

- Diese konditionalen Kommentare werden nur vom IE erfasst und ermöglichen eine zielgerichtete Einbindung
- Somit sparen wir Traffic und Berechnungszeit auf allen anderen Browsern (nicht Ziel des Skripts)
- Die Implementierung des exCanvas-Kontextes ist nahezu auf Augenhöhe mit dem original Canvas-Kontext

Referenzen:
Die Seite von excanvas [http://excanvas.sourceforge.net/]
Tutorial zur Verwendung [http://code.google.com/p/explorercanvas/wiki/Instructions]

Abwärtskompatibilität



JSON parsing - Other

*Usage stats: Global
Support: 88.63%

Method of converting JavaScript objects to JSON strings and JSON back to objects using JSON.stringify() and JSON.parse()

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser	
3 versions back	6.0	5.0	12.0	3.2	10.6	3.2		10.0		
2 versions back	7.0	6.0	13.0	4.0	11.0	4.0-4.1		11.0	2.1	
Previous version	8.0	7.0	14.0	5.0	11.1	4.2-4.3		11.1	2.2	
Current	9.0	8.0	15.0	5.1	11.5	5.0	5.0-6.0	11.5	2.3	3.0
Near future		9.0	16.0		11.6				4.0	
Farther future	10.0	10.0	17.0	6.0	12.0					

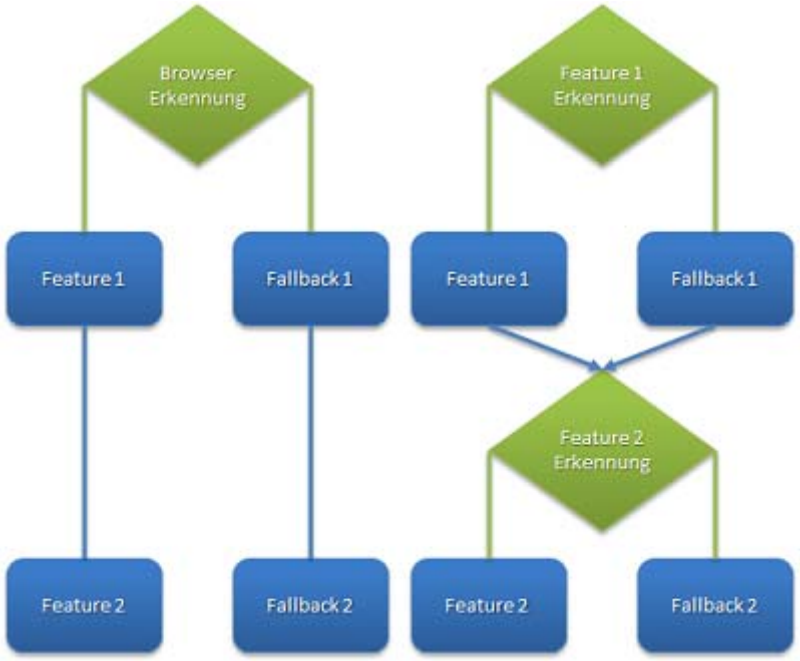
Note: Requires document to be in IE8+ standards mode to work in IE8.

Tabelle von CanIuse[1] zur Unterstützung des Datenübertragungsformats JSON in den großen Browsern

- Bisher wurde die Abwärtskompatibilität zwar angesprochen, jedoch nie kategorisiert: Wo geht was?
- Der Grund ist die Arbeitsweise des W3C und die der Browserhersteller
- Solange es sich nicht um eine **W3C Recommendation** handelt, steht eben noch nichts fest
- Es empfiehlt sich entsprechende Tabellen zu betrachten, bevor bestimmte Features eingebaut werden
- Folgende Kriterien sind daher wichtig: Welche Zielgruppe hat die Webseite? Wann wird die Webseite online gehen? Was muss die Webseite leisten können? Wieviel Arbeit (wg. Fallbacks) muss getätigt werden?

Referenzen:
Einige Tabellen zu HTML5 [http://www.findmebyip.com/litmus/]
[1] Nützliche Informationen bzgl. Kompatibilität [http://caniuse.com/#]
CSS3 Unterstützungsliste [http://www.normansblog.de/demos/browser-support-checklist-css3/]
Quirksmode Tabelle zu HTML5 Storage [http://www.quirksmode.org/dom/html5.html]
HTML5 Kompatitibilität auf mobilen Webbrowsern [http://mobilehtml5.org/]
Artikel über IE6 Unterstützung in großen Projekten [http://www.smashingmagazine.com/2011/11/03/“but-the-client-wants-ie-6-support”/]

jQuery Feature Detection



Im Gegensatz zur Browserkennung (links) hat die Feature Erkennung z.B. den Vorteil, dass Features unabhängig voneinander betrachtet werden

- Featureerkennung ist der einfachen Browsererkennung immer vorzuziehen
- Hiermit wird Browser-Diskriminierung verhindert und Kundenfreundlich gearbeitet
- Feature-Erkennung kann man z.B. mit Modernizr[1] oder anderen Bibliotheken durchführen
- Wenn jQuery bereits eingebunden ist, so kann man jQuery für einfache Erkennungen verwenden
- Alle hierzu notwendigen Eigenschaften befinden sich in `jQuery.support`, z.B.

```
01. if (jQuery.support.boxModel) { ...}
02. if (jQuery.support.cssFloat) { ... }
03. if (jQuery.support.opacity) { ... }
04. if (jQuery.support.scriptEval) { ... }
05. if (jQuery.support.hrefNormalized) { ... }
```

Referenzen:

[1] Feature Detection und viele Möglichkeiten [http://www.modernizr.com/]
MSDN Artikel zur Feature Detection [http://msdn.microsoft.com/en-us/magazine/hh475813.aspx]
jQuery Browsererkennung [http://api.jquery.com/jQuery.browser/]

jQuery Featureerkennung [http://api.jquery.com/jQuery.support/]
StackOverflow zur Feature Detektion [http://stackoverflow.com/questions/1294586/browser-detection-versus-feature-detection]

Beispiel (14)

```
01. //Filtern alle anchor-Tags
02. var hashanchors = $('a').filter(function() {
03.     //Lesen aktuellen href="" string aus
04.     var href = $(this).attr('href');
05.     //Überprüfen ob Browser normalisierte URLs unterstützt d.h. "http://meineseite/dieseseite#hash" -> "#hash"
06.     if(!$.support.hrefNormalized) {
07.         //Falls nicht, dann müssen wir den vorderen Teil ersetzen
08.         var loc = window.location; //Arbeiten über die location
09.         //Und dann über replace
10.         href = href.replace(loc.protocol + '//' + loc.host + loc.pathname, '');
11.     }
12.     //Geben zurück ob es sich um einen hash ('#' steht an ersten Stelle) handelt oder nicht
13.     return (href && href.substr(0, 1) === '#');
14. });
15. //Speichern das Ergebnis in der Konsole des Browsers
16. console.log(hashanchors);
```

- Nun könnten wir in der gefilterten Liste von <a>-Tags Manipulationen vornehmen
- So könnten wir z.B. alle Hashes entfernen und durch URLs ersetzen (hier: Speicherung in Konsole)

jQuery UI im Blickpunkt

- Manche Elemente (z.B. Range-Slider oder Datepicker) haben bis jetzt sehr geringe Unterstützung
- Des Weiteren möchte man oftmals spezielle Steuerelemente einbauen
- Beispiele hierfür: Accordions, Dialoge, Progressbars und Tabs
- Mit jQuery UI wurden all diese Optionen in einem Paket gebündelt
- Toll ist hier die sog. Themefähigkeit (über CSS) und die Verwendung mit jQuery, sowie die ausführliche Dokumentation
- Die prinzipielle Syntax baut auf jQuery auf, z.B.

```
01. $('#meindialog').dialog();//Erstellt einen Dialog - default
02. $('#print').dialog({ //Erstellt Dialog mit eigenen Optionen
03.     autoOpen: false, //Öffnet sich nicht gleich
04.     width: 600, //Größe soll 600px sein
05.     modal: true, //ist modal (d.h. besitzt immer Fokus)
06.     title: 'Druckvorschau', //Titel
07.     buttons: { "Ok" : function() {} },//Buttons mit Callbacks
08. });
```

jQuery UI ist eine Erweiterung von jQuery mit komplexen Steuerelementen

Referenzen:
Die Seite von jQuery UI [http://jqueryui.com]

jQuery UI Dokumentation [http://docs.jquery.com/Ui]
Wikipedia Artikel zu jQuery UI [http://en.wikipedia.org/wiki/JQuery_UI]
Kommerzielle Erweiterung von jQuery UI [http://wijmo.com]

jQuery UI einbinden

- Um jQuery UI einzubinden geht man in der Regel so vor wie mit jQuery
- Im Gegensatz zu jQuery ist jQuery UI auch von mindestens einer CSS Datei abhängig
- Während man die Codebasis wie schon bei jQuery über Google laden kann, wird jetzt noch eine CSS Datei benötigt
- Diese können wir auf der jQuery UI Seite herunterladen oder bei Google hosten lassen (bietet Vorteile)
- Konkret sieht das ganze dann so aus:

```
01. <!DOCTYPE html>
02. <html><head><!--Meta Tags, Titel, etc. -->
03. <link rel="stylesheet" href="https://ajax.googleapis.com/ajax/libs/jqueryui/1.8/themes/base/jquery-ui.css" />
04. </head><body><!--Inhalte etc.-->
05. <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.0/jquery.min.js"></script>
06. <script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.8.16/jquery-ui.min.js"></script>
07. <!--Eigene Skripte bzw. von jQuery oder jQuery UI abhängige Skripte-->
08. </body></html>
```

- Eine eigene Version zusammenzustellen kann auch Vorteile haben (z.B. ~26 kB gegen ~198 kB)

Referenzen:
jQuery UI bei Google-Code [http://code.google.com/apis/libraries/devguide.html#jqueryUI]
jQuery UI Webseite [http://www.jqueryui.com]
jQuery UI CSS bei Google [http://stackoverflow.com/questions/820412/downloading-jquery-css-from-googles-cdn]
How to Start Tutorial von jQuery UI [http://jqueryui.com/docs/Getting_Started]
Eigenes jQuery UI zusammenstellen [http://jqueryui.com/download]

jQuery UI Möglichkeiten

- Mit jQuery UI wird jQuery um richtige Benutzeroberflächen-Möglichkeiten erweitert
- Somit können wir für nicht vorhandene Steuerelemente (z.B. Range-Slider) einen sehr guten Fallback einbauen
- Der Vorgang würde folgendermaßen ablaufen:
    - Wieder mal bauen wir unsere Seite so auf, dass diese ohne JavaScript lauffähig ist
    - Wir verwenden HTML5 Eingabeelemente (z.B. Range-Slider, Nummerbox etc.)

    - In unserem JavaScript Code führen wir `$(document).ready(function() { ... })` aus
    - Dort fragen wir ab, ob unsere neuen Formelemente immer noch den anfangs gesetzten Wert in `type` hat
    - Falls das Attribut auf `type="text"` steht, brauchen wir einen Fallback
- Außerdem können wir unsere Webseite um notwendige Elemente wie z.B. modale Dialoge erweitern
- `draggable()` und `droppable()`
- `resizable()`
- `sortable()`
- `autocomplete()`
- `slider()`
- `accordion()`
- `datepicker()`
- `dialog()`
- `progressbar()`
- uvm.

Referenzen:
25 Tutorials zu jQuery UI [http://speckyboy.com/2010/01/20/25-tutorials-and-resources-for-learning-jquery-ui/]
Cross Browser Forms erstellen [http://net.tutsplus.com/tutorials/html-css-techniques/how-to-build-cross-browser-html5-forms/]
HTML5 Forms Fallback mit jQuery UI [http://forrst.com/posts/HTML5_Form_jQuery_fallback-hTO]
jQuery UI Widgets verstehen [http://bililite.com/blog/understanding-jquery-ui-widgets-a-tutorial/]

Neue Form Elemente für alle

Das Range-Slider Element erstellt und dargestellt mit jQuery UI unter Verwendung eines Standard Themes

- Nun haben wir festgestellt, dass wir einen Fallback brauchen - wie aufrufen?
- Prinzipiell ist jQuery UI von den Methodenaufrufen sehr ähnlich zu jQuery, d.h. wir übergeben Strings oder Objekte
- Es gibt aber noch andere Möglichkeiten wie z.B. das automatisierte Webforms2[1]
- Manche Eingabeelemente gibt es im Basic jQuery UI Paket leider nicht - häufig gibt es hierfür aber Nachbesserungen in Form von jQuery UI Plugins
- Ein Beispiel ist das `type=number` Element, welches es so nicht gibt, jedoch als das Add-on **jQuery UI Spinner** verfügbar ist
- Die beste Lösung erhält man durch Mischung verschiedener Bibliotheken
- So bietet sich z.B. Modernizr an um eine gründliche Feature Detektion durchzuführen

- Mit Webforms2 können wir Attribute wie `required`, `pattern` etc. übernehmen
- jQuery UI und Plugins helfen uns bei den einzelnen Elemente

Referenzen:

[1] Seite von Webforms2 [http://code.google.com/p/webforms2/]
Modernizr [http://www.modernizr.com/]
Seite von jQuery UI [http://www.jqueryui.com]
Das Spinner Plugin [http://plugins.jquery.com/project/jquery-ui-spinner]

Beispiel (15)

HTML Code:

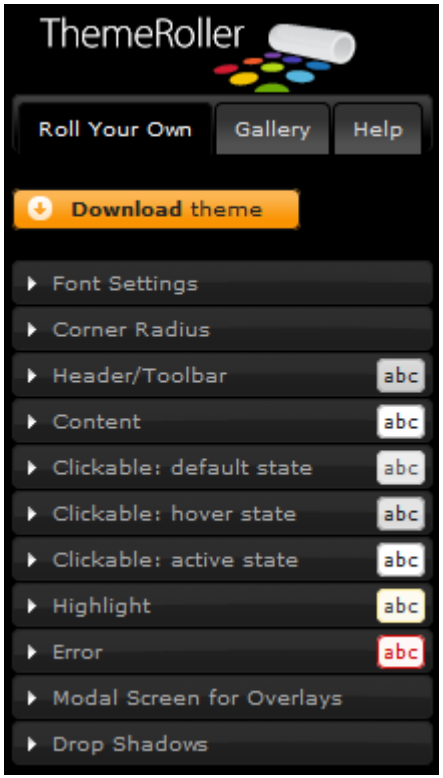
```
01. <p>Wert:<input type="range" id="ex15-range" onchange="ex15-shownumber();" value="0" />
02. <span id="ex15-shownumber"> 0 </span></p>
03. <p>Datum:<input type="datetime" id="ex15-datetime" /></p>
```

JavaScript Code:

```
01. $(document).ready(function() {//Verwenden die jQuery Ready Methode
02.     //Erstellt einen Datumsstring im Format YYYY-MM-DD mit dem aktuellen Datum
03.     document.getElementById('ex15-date').value = new Date().toISOString().substr(0,10);
04.     if(document.getElementById('ex15-range').type == 'text') //Fallback notwendig?!
05.         $('<div id="ex15-range"></div>').replaceAll('#ex15-range').slider({
06.             slide: function(event, ui) { ex15shownumber(ui.value); }//Slider-Change Event
07.         });//Beim Slider benötigen wir leider ein div - daher $().replaceAll('') notwendig
08.     if(document.getElementById('ex15-date').type == 'text')    //Beim Datepicker sieht das anders aus
09.         $('#ex15-date').datepicker({ dateFormat: 'yy-mm-dd' });
10. });//Methode um den aktuellen Wert des Sliders anzuzeigen
11. function ex15shownumber(value) { $('#ex15-shownumber').text(value); }
```

Referenzen:
Das Beispiel [http://html5skript.florian-rappl.de/uiexample/example.html]

jQuery UI Themes



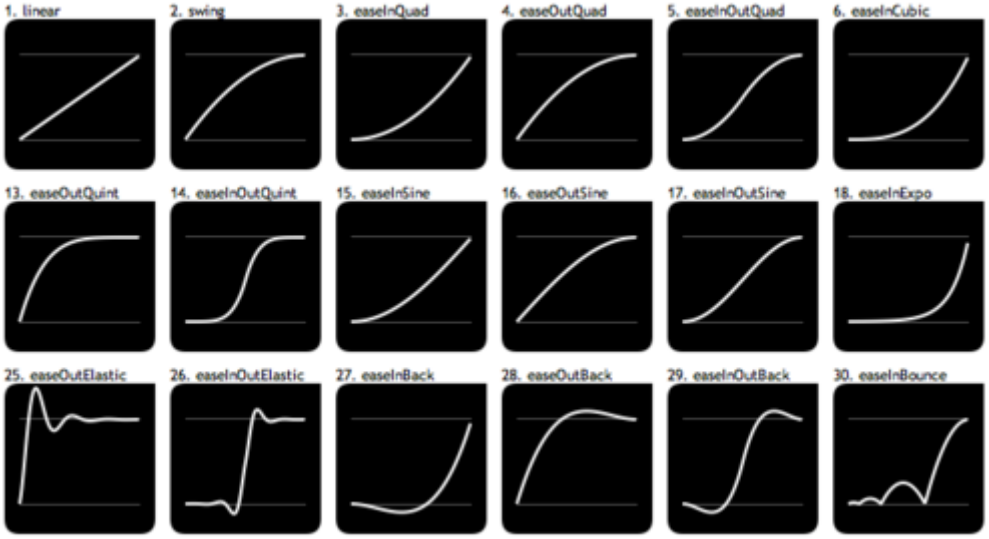
Der jQuery UI Themeroller in Aktion - in Gallery wählt man einen existierenden Theme aus, in Help findet man die Dokumentation des Themerollers

- Wie wir bereits gesehen haben ist jQuery stark abhängig vom gewählten Stylesheet (entspricht dem Theme)
- Die CSS-Datei ist auch nicht die kleinste, weshalb Änderungen eine langwierige und schwierige Aufgabe darstellen
- Dies war auch den jQuery UI Machern bekannt, daher haben diese den sog. Themeroller entwickelt
- Will man also sein eigenes Design entwickeln, so empfiehlt sich den Themeroller zu verwenden
- Der beste Weg ist daher folgender:
    1. Auswahl des Themes, der am ehesten passt
    2. Mögliche Änderungen des Themes per Themeroller
    3. Download des Themes als CSS Datei (*)
    4. Weitere notwendige Änderungen in eine eigene Datei
    5. Diese CSS Datei erst nach * einbinden (überschreiben)

Referenzen:

jQuery UI Themeroller [http://jqueryui.com/themeroller/]
O'Reilly Antwort zur Verwendung von TR [http://answers.oreilly.com/topic/1093-how-to-style-jquery-ui-widgets-with-themeroller/]
Dokumentation des Themerollers [http://jqueryui.com/docs/Theming/Themeroller]

Zusätzliche Effekte



Mit jQuery UI wird die Palette der Effekteabläufe nochmals stark erweitert

- jQuery UI erweitert auch die Palette der Effekte, die man bisher z.B. in folgendem Code verwendet hat,

```
01. $('#mydiv').animate({ /* Zu animierende Eigenschaften */} , {
02.     duration: 2000, //Zeit in ms
03.     complete: function() { }, //Was passiert nach Abschluss?!
04.     easing: 'linear' }); //Genau hier legen wir das Easing fest - in diesem Fall linear
```

- Zusätzlich kann man über `specialeasing` noch Effekte zu jeder Eigenschaft festlegen

Referenzen:
Dokumentation von `animate()` [http://api.jquery.com/animate/]
Das jQuery Easing Plugin [http://gsgd.co.uk/sandbox/jquery/easing/]
Demoseite von jQuery UI Easing [http://jqueryui.com/effect/#easing]

JavaScript Object Notation

- Wir haben bereits die grundlegende (literale) Schreibweise für ein neues Objekt kennengelernt:

```
var newObject = {};
newObject.newProperty = 2;
newObject.another = 'Hallo';
newObject.newFunction = function() {};
```

Wir können dies aber auch direkt in der Deklaration schreiben und zwar über

```
var newObject = {
    newProperty : 2,
    another : 'Hallo',
    newFunction : function() {}
};
```

Dies nennt man die JavaScript Object Notation oder kurz **JSON**

- Um ein JavaScript Objekt in Text umzwandeln kann man die Methode `JSON.stringify()` verwenden, z.B.

```
var newObject = { text : 'Hallo', title : 'Dies ist ein Beispiel', count : 2};
var stringFromObject = JSON.stringify(newObject); //String aus Objekt erstellen
console.log(newObject, stringFromObject); //Loggen Anfangswert und Resultat in der Konsole mit
```

Umgekehrt kann man über `JSON.parse()` einen String in ein JavaScript-Objekt umwandeln.

Referenzen:
Die parse-Methode bei MSDN [http://msdn.microsoft.com/en-us/library/windows/apps/cc836466(v=VS.85).aspx]
Die stringify-Methode bei MSDN [http://msdn.microsoft.com/en-us/library/windows/apps/cc836459(v=vs.85).aspx]
Wikipedia Artikel zu JSON [http://en.wikipedia.org/wiki/JSON]
Offizielle Seite mit jeder Menge Beispiele und Erklärungen [http://www.json.org/]

JSON als Datenformat

- JSON kann man auch als Datenübertragungsformat nutzen - hier muss man strikte Regeln beachten
- Abweichend von der laschen Notation in JavaScript-Code sind folgende Regeln:
  - Jeder Eintrag besteht aus einem Name-Wert-Paar, wobei der Name ein String ist
  - Strings werden immer in doppelte Anführungsstriche gesteckt - nie in einfache
  - Die Namen der Eigenschaften sind daher auch in doppelte Anführungsstriche zu stecken
  - Boolsche Werte lauten `true` und `false` und nicht `0`, `False` oder dergleichen

    - Für Arrays und Objekte gelten immer die entsprechende Literale, d.h. ein Array ist `[]`
    - Ganzzahlen, sowie Gleitkommazahlen mit oder ohne wissenschaftlicher Notation, sind erlaubt
    - Folgende Escape-Sequenzen existieren: `\"`, `\\`, `\/`, `\b`, `\f`, `\n`, `\r`, `\t` und `\uXXXX` (4 stellig Hex.)
- Natürlich gibt es für fast jede Programmiersprache bereits entsprechende Parser und Ersteller von JSON Strings
- Wenn man sich einen eigenen Serverseitigen Parser / Generator schreiben möchte, sollte man diesen vorher testen
- Für Tests auf korrekte Syntax kann man Webseiten wie JSONLint[1] nutzen

Referenzen:

[1] Seite von JSONLint [http://jsonlint.com/]
Offizielle JSON Seite [http://www.json.org]
MSDN zu JSON [http://msdn.microsoft.com/en-us/library/bb410770.aspx]
Wikipedia Artikel zu JSON-RPC [http://en.wikipedia.org/wiki/JSON-RPC]

jQuery Filter einsetzen

- Manchmal reichen die CSS Selektoren nicht aus oder man möchte die bisherige Liste verfeinern
- Hier bieten sich die sog. jQuery Filter an, z.B.
    - `eq(index)` wählt nur das Ergebnis an der (*index* + 1).-Stelle aus
    - `gt(index)` wählt Ergebnisse mit größerem Index als *index* aus
    - `lt(index)` wählt Ergebnisse mit kleinerem Index als *index* aus
    - `has('selektor')` wählt die aus die solche Elemente enthalten
    - `filter('selektor')` wendet einen weiteren jQuery Selektor an
- Es sind immer zwei Möglichkeiten vorhanden:

```
01. //Als Methode im Chaining-Konstrukt
02. var liste = $('selektor1').filter('selektor2');
03. //Als Erweiterung eines Selektors
04. var liste = $('selektor1:filter(selektor2)');
```

`filter()` kann man auch eine Funktion übergeben, die `true` (wird behalten) oder `false` (wird verworfen) zurückgibt

Durch den Einsatz von jQuery Filtern kann man Ergebnislisten weiter verfeinern

Referenzen:
jQuery Filtering [http://api.jquery.com/category/traversing/filtering/]
Tutorial über jQuery Filter [http://net.tutsplus.com/tutorials/javascript-ajax/using-jquery-to-manipulate-and-filter-data/]
Tutorial zum schnellen Filtern mit jQuery [http://kilianvalkhof.com/2010/javascript/how-to-build-a-fast-simple-list-filter-with-jquery/]
Animierter Filter für jQuery [http://webstandard.kulando.de/post/2010/02/15/quicksand-animated-jquery-filter]

Beispiel (16)

```
01. //Basisauswahl erstellen
02. var baseList = $('article > section');
```

```
03. console.log('baseList', baseList);//Loggen
04. var filter1 = baseList.has('pre');
05. //Die Basisauswahl filtern (1)
06. console.log('filter1', filter1);//Loggen
07. //Die Basisauswahl filtern (2)
08. var filter2 = baseList.eq(7);
09. console.log('filter2', filter2);//Loggen
10. //Die Basisauswahl filern (3)
11. var filter3 = baseList.filter('#lec08-example1');
12. console.log('filter3', filter3);//Loggen
13. //Die gefilterte (1) Basisauswahl filtern (3)
14. var filter1_3 = filter1.filter('#lec08-example1');
15. console.log('filter1_3', filter1_3);//Loggen
16. //Konkretes Beispiel
17. $('body > article > section') //Verwenden eine Basisliste
18.     .css('background', '#F00') //alles mit rotem Hintergrund darin!
19.     .filter('#lec08-example1') //Filtern die Basisliste
20.     .css('background', '#0F0'); //Setzen den Hintergrund auf Grün in der gefilterten Liste
```

jQuery Utilities nutzen

- jQuery bietet uns jede Menge nützlicher Methoden für die tägliche Arbeit mit JavaScript
- So haben wir bereits `$.each()` und `$.support` kennengelernt
- Mit jQuery ist auch die Browsererkennung einfacher: `$.browser` enthält alle Informationen
- Über `$.extend(target, default)` kann man ein Objekt mit *default*–Werten erweitern
- Mit `$.grep(array, function(v, i) { ... })` kann man Arrays sehr leicht filtern
- Durch `$.merge(first, second)` werden zwei Arrays zu einem neuen Array zusammengefügt, z.B.

  ```
  01. var copyOfArray = $.merge([], oldArray); //Erstellt Kopie von oldArray in copyOfArray
  ```

  Vorsicht: Die einzelnen Objekte sind u.u. immer noch referenziert, nur das Array ist kopiert.
- `$.map(array, function(v, i) { ... })` weist jedem Wert eines Arrays einen neuen Wert zu, z.B.

```
01. var squares = $.map( [2, 3, 4, 5, 6], function(n) { //Iteriert über das Array
02.    return n * n; //Quadriert den Weg des Eintrags und gibt diesen Wert zurück
03. }); //squares = [4, 9, 16, 25, 36]
```

Vorsicht, da `$.each()` als Parameter zuerst den *index* und dann den *Wert* verwendet (hier umgekehrt)!

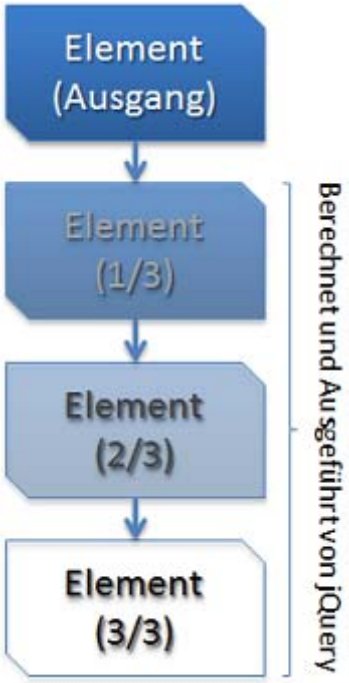
Referenzen:
jQuery Utilities [http://api.jquery.com/category/utilities/]
jQuery Browser Detektion im Detail [http://msmvps.com/blogs/luisabreu/archive/2009/08/07/jquery-utilities-functions-i-browser-detection.aspx]
Einführung zu den Utilities [http://www.tutorialspoint.com/jquery/jquery-utilities.htm]
Utilities auf jQuery4U [http://www.jquery4u.com/category/utilities/]

Mehr Animationen!

- Bis jetzt haben wir uns nur die sehr allgemeine `animation()` Methode angeschaut
- In jQuery sind nicht nur einige Effekte (z.B. `linear`) bereits eingebaut, sondern auch einige Animationen
- Über die Methoden `fadeIn()`, `fadeOut()` und `fadeTo()` kann man Elemente sichtbar(er) bzw. unsichtbar(er) machen
- Analog kann man über `slideDown()` und `slideUp()` Elemente sichtbar bzw. unsichtbar machen
- Dazu gibt es noch über `fadeToggle()` bzw. `slideToggle()` die Möglichkeit zwischen Zuständen zu wechseln, z.B.

```
01. $('#clickme').click(function() { //Was passiert beim Klick?
02.   $('#book').slideToggle('slow', function() { //Animation
03.     //Callback; 'slow' ist ein Macro von jQuery mit 600 ms
04. }); }); //'normal' entspricht 400 ms, 'fast' 200 ms
```

Zusätzlich kann man über `show()`, `hide()` und `toggle()` durch Übergabe einer Zeitangabe in ms auch animieren



Mit Hilfe der eingebauten fadeout-Animation kann man Elemente sehr elegant verstecken

Referenzen:
jQuery Effekte [http://api.jquery.com/category/effects/]
Tutorial zu den toggle() Methoden [http://www.sohtanaka.com/web-design/easy-toggle-jquery-tutorial/]
Tutorial zu Animationen [http://www.randomsnippets.com/2011/04/10/how-to-hide-show-or-toggle-your-div-with-jquery/]
W3Schools zu jQuery-Toggle [http://www.w3schools.com/jquery/eff_toggle.asp]

Beispiel (17)

```
01. function onChangeHandler(s) {
02.     var value = 'slide'; //Wert der ausgewählten Radiobox rausfinden und in value speichern - Standardwert slide
03.     $('input:radio[name=radiogroup]').each(function(i,v) { //Alle RadioBoxen die "radiogroup" als name=... haben
04.         if(v.checked) //Wenn die angewählt ist
05.             value = v.value; //Den Namen speichern
06.     });
07.     var selection = s.value; //Der Wert der Nummernbox in selection abspeichern (ist String!)
08.     function filterList(i, v) { //Unsere Methode um die Liste zu filtern - wenn der Text der selection entspricht
09.         return $(v).text() === selection; //Dann bleibt das betrachtete Element in der Liste - ansonsten nicht
```

```
10.     };
11.     if(value === 'slide') //Slide ausgewählt?
12.         $('.box').slideUp('fast').filter(filterList).slideDown('slow');
13.     else if(value === 'fade') //Fade ausgewählt?
14.         $('.box').fadeOut('fast').filter(filterList).fadeIn('slow'); //Slow ist 600 ms
15.     else //Toggle ausgewählt?
16.         $('.box').hide('fast').filter(filterList).show('slow'); //Fast ist 200 ms
17. }
18. $(document).ready(function() { //Verstecken die (sehr) langsam am Anfang
19.     $('.box').fadeOut(2000); //über einen FadeOut Effekt mit 2000 ms
20. });
```

Was ist AJAX?

Mit AJAX können Inhalte direkt auf einer Seite untergebracht werden und somit Ladezeiten sowie Verzögerungen durch Seitenaufbau verringert werden

- AJAX steht für Asynchronous Javascript And XML – damit sind zwei Technologien gemeint:
  - Asynchroner JavaScript, d.h. eine Anweisung die **nicht-blockend** ist und auf ein Ergebnis **wartet**
  - Das Ergebnis kann in Form eines XML-Dokuments, welches **geparst** werden kann, vorliegen
- Die Vorteile von AJAX liegen auf der Hand und schaffen auch die Basis für den Erfolg einiger Webseiten
- Ein Problem in der Verwendung bestand darin, dass Browser unterschiedlich behandelt werden müssen
- jQuery erledigt diese nervige Aufgabe unter anderem für uns – außerdem parst jQuery die erhaltenen Daten und gibt uns sofort verwendbare Daten zurück

Referenzen:
Wikipedia Artikel zu AJAX [http://de.wikipedia.org/wiki/Ajax_(Programmierung)]
W3Schools über AJAX [http://www.w3schools.com/Ajax/ajax_intro.asp]
About AJAX [http://webtrends.about.com/od/web20/a/what-is-ajax.htm]
AJAX Tutorial von W3Schools [http://www.w3schools.com/ajax/default.asp]

AJAX mit jQuery

- Alle AJAX Abfragen laufen in jQuery über `$.ajax()`, auch speziellere Methoden wie `$.getJSON()`
- jQuery bietet uns standardmäßig Ereignisse und Möglichkeiten die Abfrage individuell zu gestalten
- Die grundsätzliche Syntax lautet

```
01. var settings = {}; //Einstellungen für die Abfrage
02. var ajaxObjekt = $.ajax('url der abfrage', settings); //Abfrage tätigen
03. ajaxObjekt.abort(); //Abfrage abbrechen, optional wenn notwendig
```

  die zurückgegebene Variable ist optional - mit ihr ist es z.B. möglich den Vorgang abzubrechen
- Fast alle Möglichkeiten stecken in dem übergebenen `settings`-Objekt, z.B.

```
01. var settings = {
02.     async: true, // könnten den Request auch Blockend aufbauen - mit false
03.     cache: false, // jQuery sendet in der URL ein _=[TIMESTAMP] mit um Caching zu verhindern
04.     complete: function(obj, status) { }, // Callback wenn abgeschlossen
05.     crossDomain: false, // aus Sicherheitsgründen notwendig - true wenn Zugriff auf anderen Server
06.     data: { }, // Objekt das an den Server mittels Name-Wert-Paaren geschickt wird (a la HTML-Form)
07.     dataType: 'xml', // Datentyp des AJAX-Requests - auch JSON, Script bzw. HTML sind möglich
08.     error: function(obj, status, err) { }, // Callback wenn Fehler auftritt
09.     success: function(data, status, obj) { }, // Callback wenn erfolgreich abgeschlossen
10.     type: 'GET' } // bestimmt den Typ des Requests - auch POST, PUT etc. sind möglich
```

Referenzen:
jQuery AJAX [http://api.jquery.com/jQuery.ajax/]
PHP AJAX Login mit jQuery [http://www.chazzuka.com/php-ajax-login-form-using-jquery-82/]
Tutorial AJAX mit jQuery und PHP [http://www.sitepoint.com/ajax-jquery/]
Noch ein jQuery AJAX mit PHP Tutorial [http://www.php4every1.com/tutorials/jquery-ajax-tutorial/]

Beispiel (18)

```
01. //Setzen des Ladebildes
02. $('#example18insert').html('<img src="loader.gif" alt="Ladebalken für AJAX" />');
03. //Ausführen von AJAX
04. $.ajax({
05.     url: 'http://html5.florian-rappl.de/submitted.html', //Wir fragen diese (statische) Seite ab
06.     cache: false, //Es soll nicht gechached werden -- wollen diese Seite immer neu erhalten
07.     dataType: 'html', //Setzen HTML als Datentyp fest
08.     type: 'GET', //Setzen GET als Methode fest
09.     success: function(html, status) { //Was passiert wenn alles gepasst hat?
10.         console.log('Erfolg.');
11.         setTimeout(function() { $('#example18insert').html(html); }, 1000); //Content setzen
12.     },
13.     error: function(xhr, status, err) { //Was passiert im Falle eines Fehlers?
14.         $('#example18insert').html(''); //Content zurücksetzen
15.         alert('Fehler in der Abfrage!');
16.     },
17.     complete: function(xhr, status) { console.log('Abgeschlossen.'); } //Was passiert zum Abschluss?
18. });
```

Das richtige Datenformat

- Eine wichtige Entscheidung ist die Wahl des Datenformats. Folgende Möglichkeiten existieren:
  - HTML: Naheliegend und besitzt durchaus Vorteile wenn Teile der Seite komplett ersetzen wollen. Allerdings sind Auswertungen direkt nur schwer möglich sind und großer Overhead inbegriffen.
  - XML: Aufgrund des X in AJAX ein beliebtes Format, u.a. weil es das Standardaustauschformat von Webservices (SOAP) ist. Dokumente sind sehr schnell (vom Browser) geparsed. Der Nachteil ist, dass es zum direkten Anzeigen schlechter als HTML geeignet ist und noch immer zu viel Overhead besitzt.
  - JSON: Von allen Formaten am schlechtesten zur direkten Anzeige zu gebrauchen – allerdings auch mit dem geringsten Overhead. Der Hauptvorteil liegt darin, dass JSON fast natives JavaScript ist und daher die Auswertung am schnellsten durchgeführt werden kann.

```
01. <data>
02.     <seasons>
03.         <season>Winter</season>
04.         <season>Sommer</season>
```

```
    </seasons>
    <days>
        <day name="Montag" short="Mo" abbr="M"/>
        <day name="Dienstag" short="Di" abbr="D"/>
        <day name="Mittwoch" short="Mi" abbr="M"/>
    </days>
</data>
```

```
{
    "seasons":[
        "Winter",
        "Sommer"
    ],
    "days":[
        {"name":"Montag","short":"Mo","abbr":"M"},
        {"name":"Dienstag","short":"Di","abbr":"D"},
        {"name":"Mittwoch","short":"Mi","abbr":"M"}
    ]
}
```

Referenzen:
Quirksmode über das richtige Format [http://www.quirksmode.org/blog/archives/2005/12/the_ajax_respon.html]
JSON gegen XML für Webservices [http://digitalbazaar.com/2010/11/22/json-vs-xml/]
Debatte JSON / XML [http://ajaxian.com/archives/json-vs-xml-the-debate]
Vergleich von JSON und XML Parsern [http://blogs.nitobi.com/dave/2005/09/29/javascript-benchmarking-iv-json-revisited/]
Harvard Vorlesung zu Webentwicklung mit XML [http://cscie153.dce.harvard.edu/lecture_notes/2009/20091027/handout.html]

Grenzenlos austauschen

- Womit man sehr schnell Probleme kriegen wird sind AJAX-Abfragen über den Server hinweg
- Mit Server ist der Rechnername von dem die Seite stammt gemeint (z.B. *html5skript.florian-rappl.de*)
- Das Problem liegt in der Gefahr sog. Cross-Site Scripting, d.h. gefährlichen externen Programmen
- Ein möglich Ausweg ist serverseitig[1] - ein Skript auf dem Server führt die externe Abfrage durch
- Ein geschickterer Weg läuft über eine sog. JSONP-Abfrage - hier wird ein `<script>`-Element erstellt

- Dieses Element setzt das `src`-Attribut auf die aktuelle Abfrage, welche dann eine von uns erstellte Methode mit dem JSON Objekt (eigentliche Rückgabe) aufruft, z.B.

```
<script>
function parseResponse(obj) {
    // Mach was mit dem JSON Objekt genannt obj
}
</script>
<script src="http://extern.example/request?UserId=1234&jsonp=parseResponse"></script>
```

- Natürlich ist dieser Weg bereits in jQuery eingebaut, weshalb wir uns keine solche Methode erstellen müssen

Referenzen:
JSONP bei Wikipedia [http://en.wikipedia.org/wiki/JSONP]
Remy Sharp zu JSONP [http://remysharp.com/2007/10/08/what-is-jsonp/]
[1] Einfaches PHP Script zum grenzenlosen Austausch [http://jquery-howto.blogspot.com/2009/04/cross-domain-ajax-querying-with-jquery.html]
Die offizielle Seite [http://json-p.org/]

Beispiel (19)

```
//Wieder setzen wir das Ladebild zunächst
$('#example19insert').html('<img src="loader.gif" alt="Ladebalken für AJAX" />');
//Nun starten wir die JSON Abfrage
$.ajax({ //Als URL (Beispiel) nehmen wir die serverseitige Suche - entscheidend ist das callback=?
    url: 'http://html5skript.florian-rappl.de/search.aspx?callback=?',
    cache: false,//Wieder wird nicht gecached
    type: 'GET', //Und wir machen abermals GET
    dataType: 'json', //Hier wählen wir JSON - durch callback=? wird eine JSONP Abfrage erzeugt
    data: { s : $('#example19text').val() }, //Als Daten übergeben wir den Inhalt der Textbox
    success: function(data) {
        var content = []; //Sammeln das Ergebnis in einem Ausgabearray
        $.each(data.hits, function(i,v) {
            content.push(v.title + ' (' + v.url + ')');
        });
        $('#example19insert').html(content.join('<br/>')); //Das durch join mit Zeilenumbrüchen verbunden wird
        $('#example19count').html(data.hits.length + ' Einträge gefunden.');//Geben auch Anzahl d. Ergebnisse aus
```

```
17.         console.log(data);//Hier sehen wir was wir wirklich gekriegt haben (bereits mit jQuery bearbeitet)
18.     }
19. });
```

Was kann jQuery noch?

- Damit man sich viel Schreibarbeit sparen kann ist es möglich die Standardeinstellung von jQuery's AJAX Methode zu ändern,

```
01. $.ajaxSetup({
02.     cache: true, //Soll immer gecached werden
03.     dataType: 'JSON', //Soll immer als JSON erwartet werden
04.     error: function() { ... }, //Soll immer hier abgefangen werden
05.     timeout: 60000, //Immer 60 Sekunden Timeout - also 1 Minute
06.     type: 'POST' }); //Standardmäßig POST verwenden
```

  Somit können wir uns das Einstellen dieser Eigenschaften sparen

- Des Weiteren erlaubt uns jQuery Elemente über Ereignisse an AJAX zu koppeln, z.B.

```
01. $('#irgendwas').ajaxError(function() {
02.     /* Hier steht unser Code - wird aufgerufen wenn ein AJAX Request fehlschlägt */ });
03. $('#wasanders').ajaxSend(function() {
04.     /* Hier steht unser Code - wird VOR jedem Sendevorgang aufgerufen */ });
```

  Weitere Ereignisse sind `ajaxStart`, `ajaxSuccess`, `ajaxComplete` und `ajaxStop`

- Es gibt spezialisierte Methoden, die `ajax()` mit entsprechenden Parametern aufrufen, z.B. `getScript()`, `getJSON()`, `load()`, `get()` und `post()`

Referenzen:
Die Verwendung von ajaxSetup [http://www.bennadel.com/blog/2131-Using-jQuery-ajaxSetup-To-Accumulate-Global-Data-Parameters-For-AJAX-Requests.htm]
Tutorialspoint über ajaxSetup [http://www.tutorialspoint.com/jquery/ajax-jquery-ajaxsetup.htm]
jQuery ajaxSetup Dokumentation [http://api.jquery.com/jQuery.ajaxSetup/]
Alle jQuery AJAX Methoden [http://api.jquery.com/category/ajax/]

Die Messaging API

- Die Messaging API erfreut sich optimaler Browserunterstützung und bietet uns eine einfache Möglichkeit Text zwischen Seiten unterschiedlicher Herkunft zu versenden oder zu empfangen
- Im Regelfall wird die Messaging API dazu verwendet, dass Webseiten miteinander kommunizieren

- Normalerweise wird die Seite mit der man kommunizieren möchte in einem neuen Fenster oder `<iframe>` geladen – und zwar von unserer Seite aus, da wir sonst keinen Zugriff haben
- Wichtig ist, dass wir eine Referenz auf das `window` Objekt dieser Seite erhalten
- Weiterhin muss eine Nachricht natürlich aufgenommen werden können
- Dies kann man durch eine Ereignisbehandlung auf das `message` Ereignis ermöglichen, z.B.

```
01. window.addEventListener('message', function(event) {
02.     //event.data gibt uns die Nachricht an
03.     //event.origin gibt uns die URL (der sendenden Seite) an
04.     //event.source gibt uns eine Referenz zum window-Objekt der Quelle
05.     event.source.postMessage('Howdy Cowboy!', event.origin); //Schreiben eine Nachricht zurück
06. }, false); //Wir wollen nur Message Ereignisse an das window-Objekt behandeln
```

- Somit kann man sehr leicht Kommunikationsschnittstellen in einer Webseite einbauen oder ausnutzen

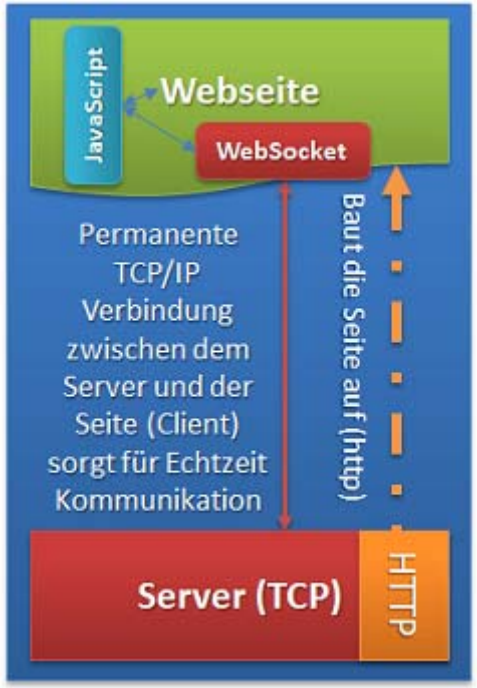
Referenzen:
W3C Spezifikation [http://www.w3.org/TR/webmessaging/]
Tutorial zur Messaging API [http://marakana.com/bookshelf/html5_tutorial/messaging.html]
MDN zu postMessage [https://developer.mozilla.org/en/DOM/window.postMessage]
Demo zu Messaging API [http://html5demos.com/postmessage2]

Die Zukunft: WebSockets!

- Es wird eine ständige **TCP**-Verbindung zum Server hergestellt
- Dadurch erhält man gegenüber AJAX mehrere Vorteile:
  - Direkter Zugang zum Server – keine Beschränkung mehr auf http
  - Viel weniger Overhead (User-Agent, Content-Type, ...)
  - Kein neues Erstellen von Objekten inkl. Öffnen und Schließen der Verbindung notwendig
- Die Syntax ist sehr einfach (ganz im Gegensatz zu echter Socket-Programmierung) und analog zur Messaging API:

```
01. var socket = new WebSocket('ws://example:8080'); //Erstellen
02. socket.onmessage = function(event) { //Beim Eintreffen
03.     alert(event.data); }; //Nachricht als Msg-Box
04. socket.onclose = function() { //Beim Schließen
05.     socket.send('See you Dudes!'); }; //Senden
06. socket.onopen = function() { //Beim Öffnen
```

```
07.     socket.send('Hiyo!'); }; //Senden
```



Im Gegensatz zu AJAX läuft die Kommunikation bei Websockets in beide Richtungen und findet direkt auf TCP/IP Basis anstelle des HTTP-Headers statt

- Für Ultra-Responsive Applications (z.B. Multiplayer Spiele) sind die WebSockets daher essentiell

Referenzen:
Tutorial um Node.js auf einem Server einzurichten [http://remysharp.com/slicehost-nodejs-websockets/]
Wikipedia über WebSocket [http://en.wikipedia.org/wiki/WebSocket]
W3C Spezifikation zu Websockets [http://dev.w3.org/html5/websockets/]
Websocket Demo [http://html5demos.com/web-socket]

JavaScript Promises

- Durch Promises ist es möglich auf ein Ergebnis zu warten ohne dass diese Wartezeit die Seite blockiert
- Das Konzept wurde von Microsoft wg. einer ähnlichen Technologie in .NET 4.5 eingeführt
- Man verringert die Zahl der notwendigen Callbacks und schafft mehr Übersichtlichkeit
- Außerdem kann man durch dieses Konzept wieder **Chaining** sehr gut nutzen
- Ein JavaScript Promise ist definiert durch folgende Zustände:

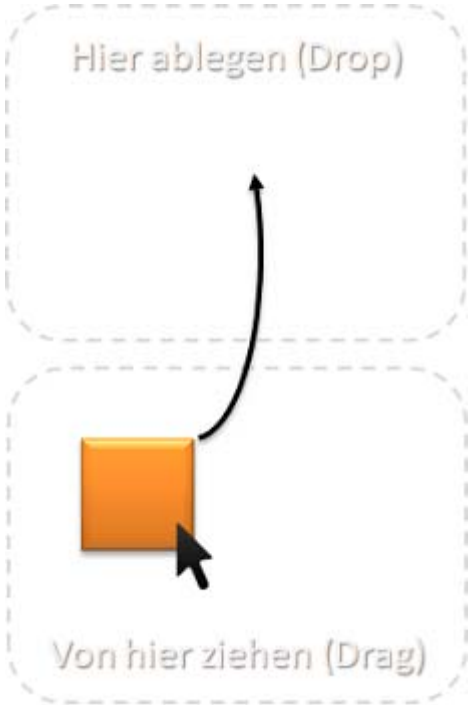
    - Zustandslos
    - Erfolgreich
    - Fehlgeschlagen
- Der Code einer asynchronen (d.h. einer Methode die länger als 50 ms benötigt und nicht-blockend sein soll) Methode würde nun so ausgewertet werden

```
01. function doAjax() { return $.get('foo.htm'); } //Blockende AJAX Methode
02. function doMoreAjax() { return $.get('bar.htm'); } //Noch eine andere blockende AJAX Methode
03. $.when( doAjax(), doMoreAjax() ) //Dies sind unsere langsamen (aber notwendigen) Methoden => ausgelagert
04.     .then(function() { console.log( 'I fire once BOTH ajax requests have completed!' ); }) //Erfolg
05.     .fail(function() { console.log( 'I fire if one or more requests failed.' ); }); //Misserfolg
```

Referenzen:
MSDN zu Promises [http://msdn.microsoft.com/en-us/library/windows/apps/hh464930.aspx]
Erklärung der Verbesserung mit Promises [http://blogs.msdn.com/b/ie/archive/2011/09/11/asynchronous-programming-in-javascript-with-promises.aspx]
Promises mit jQuery [http://www.slideshare.net/ryan_blunden/promises-in-java-script-with-jquerykey]
Deferreds in jQuery nutzen [http://www.erichynds.com/jquery/using-deferreds-in-jquery/]

Drag and Drop

- Die Drag-And-Drop API kommt vom IE(5) - sie wurde für HTML5 Reverse Engineered
- Alle Browser sollten die API mittlerweile im Prinzip unterstützen
- Die API gilt leider als eine der schlechteren - hat aber durch seine Verbreitung Vorteile
- Im Fokus stand die Standardisierung des Drag and Drop Vorgangs
- Standardmäßig `draggable` sind Text, Bilder und Links
- Um solche Elemente ablegen zu können müssen Ereignisse in den Ablagezonen gesetzt werden
- Über `return false` in `ondrop` erlauben wir das Ablegen



Dank Drag and Drop ist es möglich Objekte direkt miteinander interagieren zu lassen

- Beim IE muss man hier noch `ondragenter` und `ondragover` (analog zu `ondrop`) setzen
- `ondragover` gibt uns die Chance bei Ablage etwas mit den Daten anzufangen, z.B. mit Text über

```
01. element.ondrop = function(event) { var text = event.dataTransfer.getData('Text'); }
```

Referenzen:
Die W3C Spezifikation [http://dev.w3.org/html5/spec/editing.html#dnd]
Reverse Engineering der Original API [http://ln.hixie.ch/?start=%201115899732&count=1]
YouTube Video über Drag and Drop [http://www.youtube.com/watch?v=Tg6okHlWwv0]
Beispiele von IntroducingHTML5 [http://introducinghtml5.com/examples/ch08/]

Raus aus dem Browser

- Es ist möglich Interaktionen zwischen dem Browser und anderen Applikationen über die API zu beschreiben
- Das Objekt `dataTransfer`, das wir mit einem DnD-Ereignis erhalten enthält Methoden:
  - `getData(mime)` mit einem Mime-Type z.B. `'text/plain'`
  - `setData(mime, data)` mit einem zusätzlichen String von Daten

- Ein Problem ist, dass man sich schon bei `ondragstart` entscheiden muss, was man für Daten setzt
- Dies bedeutet, dass man nicht auf erst beim Ablegen (Worauf wird abgelegt?) entscheiden kann
- Alles in allem hat man am Ende sieben Ereignisse mit `ondragend`, `ondrag` und `ondragleave`
- Neben den erwähnten Elementen kann in HTML5 jedes Element gezogen werden
- Hierzu benötigt man das Attribut `draggable` (bzw. `draggable="true"`)
- Bei älteren Safaris wird daneben noch eine CSS-Regel benötigt:

```
01. [draggable] { -webkit-user-drag: element; }
```

- Ein großes Problem ist leider die Cross-Browser-Unterstützung...

Referenzen:
Beispiele von IntroducingHTML5 [http://introducinghtml5.com/examples/ch08/]
Remy Sharp über die DnD API [http://html5doctor.com/native-drag-and-drop/]
Quirksmode über das DnD Desaster [http://www.quirksmode.org/blog/archives/2009/09/the_html5_drag.html]
DnD File Upload Tutorial [http://www.thebuzzmedia.com/html5-drag-and-drop-and-file-api-tutorial/]
Ausführlicher Artikel über DnD [http://www.html5laboratory.com/drag-and-drop.php]
MDN zur Drag and Drop API [https://developer.mozilla.org/en/DragDrop/Drag_and_Drop]

DND Fallback

- IE unterstützt erst ab IE 10 die aktuelle API (bisher: ältere), Opera frühestens ab Version 12 (bisher: keine)
- Die Frage ist also: Welche Möglichkeiten hat man neben der offiziellen API?
- Hier hilft uns wieder jQuery UI - wir können Elemente nun über JavaScript Drag- und Dropfähig machen
- Die Syntax ist dabei ziemlich einfach und straight-forward:

```
01. $(function() {
02.     $("#draggable").draggable();
03.     $("#droppable").droppable({
04.         hoverClass: 'dropOver',
05.         accept: '#draggable',
06.         //Event von ein Element (innerhalb der Toleranzen) reingezogen wurde
07.         drop: function(event, ui) {
08.             //this is das drpppable Element
09.             //ui.draggable das draggable Element
10.             $(this).text("Es wurde folgendes gedroppt: " + $(ui.draggable).text());
```

```
        }
    });
});
```

- Mit `accept` stellt die akzeptierten Elemente ein, `hoverClass` legt eine CSS-Klasse für den Hovereffekt fest

Referenzen:
jQuery UI Demo zu Draggable [http://jqueryui.com/demos/draggable/]
jQuery UI Demo zu Droppable [http://jqueryui.com/demos/droppable/]
30 coole Demos zu jQuery und DnD [http://www.1stwebdesigner.com/freebies/drag-drop-jquery-plugins/]
Ausführliches Tutorial zu jQuery UI DnD [http://www.elated.com/articles/drag-and-drop-with-jquery-your-essential-guide/]

Beispiel (20)

```
//Ein Dragicon Bild erstellen (wird nicht angezeigt)
var dragIcon = document.createElement('img');//Bildelement erstellen
dragIcon.src = 'icons/me.png';
var elements = document.querySelectorAll('.ex20-draggable');
//Was passiert wenn die einzelnen Elemente gezogen werden?
for(var i in elements) //So erhalten wir alle Indizes
    elements[i].ondragstart = function(event) { //Das ist wichtig!
        event.dataTransfer.setData('text/plain', this.id);//Daten setzen
        event.dataTransfer.setDragImage(dragIcon, 10, 33);//Setzen Icon
    };
//Events festlegen - jede Menge Ereignisse für die 2 Boxen
document.querySelector('#ex20-dragtarget').ondrop = dropSomething;
document.querySelector('#ex20-dragsource').ondrop = dropSomething;
document.querySelector('#ex20-dragtarget').ondragover = acceptDrop;
document.querySelector('#ex20-dragtarget').ondragenter = acceptDrop;
document.querySelector('#ex20-dragsource').ondragover = acceptDrop;
document.querySelector('#ex20-dragsource').ondragenter = acceptDrop;
//Funktionen festlegen -- beide sind sehr wichtig
function dropSomething(event) { //Nehmen das Eventarg. mit
    var id = event.dataTransfer.getData('text/plain');//Daten auslesen
    this.appendChild(document.querySelector('#' + id));//Element anh.
```

```
22.     return false; }
23. //Machen hier keine Unterscheidung - akzeptieren den Drop immer!
24. function acceptDrop(event) { return false; }
```

Daten speichern - Überblick

- Beim clientseitigen Speichern von Daten kann man prinzipiell zwei Unterschiedliche Arten festhalten:
  - Flüchtige Speicherung (Nur solange der Browser läuft)
  - Permanente Speicherung (Wird i.d.R. auch beim nächsten Browserstart noch vorhanden sein)
- In der 1. Kategorie gab es die sog. Sessions, in der 2. die Cookies
- Eine Session ist jedoch nichts anderes wie ein Cookie ohne Ablaufdatum (daher nur gültig bis der Browser geschlossen wird)
- In HTML5 sind noch zwei weitere Möglichkeiten hinzugekommen:
  - WebStorage um eine effizienterre Zugriffsmöglichkeit zu schaffen (für 1. und 2. Kategorie)
  - WebSQL um eine integrierte Datenbank einzubauen (nur für 2. Kategorie)
- Im Gegensatz zu Cookies (und damit auch Sessions) sind die neuen Technologien nur Clientseitig verfügbar
- Bei WebSQL ist das letzte Wort noch nicht gefallen - mittlerweile geht es eher in die Richtung IndexedDB bzw. ehemals WebSimpleDB
- Fakt ist, dass es bald eine (auf allen Browsern laufende) per SQL anzusprechende Datenquelle gibt

Referenzen:
W3C Spezifikation zu WebStorage [http://dev.w3.org/html5/webstorage/]
W3C Spezifikation zu IndexedDB [http://www.w3.org/TR/IndexedDB/]
Artikel über die neuen Technologien [http://www.dev-articles.com/article/Html-5-part-3-:-Data-Storage-423001]
LocalStorage jetzt schon verwenden [http://www.webmonkey.com/2011/04/how-to-use-html5s-local-storage-tools-today/]

Das Cookie

Cookies sind eine nützliche, aber auch gefährliche Technologie – außerdem sind sie nicht gerade Entwicklerfreundlich

- Über JavaScript hat man durch `document.cookie` Zugriff auf die Cookies
- Da der Zugriff nicht straight-forward ist empfiehlt es sich Peter-Paul Kochs Skript[1] zu verwenden
- Restriktionen gibt es aber überall: Nur **4 kB** pro Cookie und nur **Texte** können gespeichert werden
- Cookies dienen dazu Informationen *zwischen* Seiten auszutauschen
- Ein Cookie besteht immer aus einem Name-Wert Paar (maximal 20 davon)
- Daneben werden noch Attribute wie Domäne, Pfad, Ablaufdatum (oder max. Alter), sowie Sicherheit (z.B. Secure oder httpOnly) eingestellt
- Die Browser schicken die Attribute nicht an Server

- Durch Cross-Site-Scripting und Abhören von Verbindungen sind Cookiediebstähle etc. möglich

Referenzen:
Cookies bei W3Schools [http://www.w3schools.com/js/js_cookies.asp]
Wikipedia Artikel [http://en.wikipedia.org/wiki/HTTP_cookie]
[1]Quirksmode Artikel zu Cookies [http://www.quirksmode.org/js/cookies.html]
Artikel zu Cookies bei HTMLGoodies [http://www.htmlgoodies.com/beyond/javascript/article.php/3470821/So-You-Want-To-Set-A-Cookie-Huh.htm]
GitHub MooTools Quellcode Cookiemodul [https://github.com/mootools/mootools-core/wiki/Cookie]

Die Session

- Die Session ist ein Spezialfall eines Cookies - ein Cookie ohne Ablaufdatum
- Die Session funktioniert daher nicht bei deaktivierten Cookies - hier muss man sich andere Wege überlegen
- Die Aufgabe der Session besteht im wesentlich darin Benutzer eine Programminstanz vorzugaukeln
- Die Problemstellung liegt darin, dass Webserver instanzlos agieren (d.h. Benutzerinfos vergessen)
- Daher muss man das bisher gemachte (z.B. Artikel in einen Warenkorb gelegt) auf dem Server speichern und eine ID für den Datensatz festlegen; für eine richtige Zuordnung muss sich der Client mit dieser ID identifizieren
- Beispiel des Erstellens und Auslesens einer Session in JavaScript:

```
01. function createSession(name, value) { //Funktion zum Erstellen einer Sessionvariable
02.     document.cookie = name + "=" + value + "; path=/"; //Kein Ablaufdatum etc.
03. }
04. function readSession(name) { //Funktion zum Auslesen einer Sessionvariable
05.     var ca = document.cookie.split(';'); //Immer am Strichpunkt aufspalten
06.     for(var i = 0; i < ca.length; i++) { //Über alle Unterelemente iterieren
07.         while (ca[i].charAt(0) == ' ') ca[i] = ca[i].substring(1, ca[i].length); //Leerzeichen entfernen
08.         if (ca[i].indexOf(name + "=") == 0) return ca[i].substring(name.length + 1, ca[i].length); }
09. }
```

Referenzen:
MSDN zu Sessions [http://msdn.microsoft.com/en-us/library/windows/desktop/aa384322(v=vs.85).aspx]
Wikipedia Artikel [http://en.wikipedia.org/wiki/Session_(computer_science)]
Doug Lea über das Session Problem [http://g.oswego.edu/dl/pats/session.html]
Peter-Paul Kochs Cookieskript [http://www.quirksmode.org/js/cookies.html]

Beispiel (21)

```
01. function createCookie(name, value, days) { /* Das Cookie erstellen */
02.     var expires = ''; //Wichtig ist hier auf das Ablaufdatum zu achten
03.     if (days) {    //Bei days === 0 haben wir eine Session!
04.         var date = new Date();
05.         date.setTime(date.getTime() + (days * 24 * 60 * 60 * 1000)); //d * (h/d) * (min/h) * (s/min) * (ms/s)
06.         expires = '; expires=' + date.toGMTString(); // So wird das Ablaufdatum gesetzt
07.     }
08.     document.cookie = name + '=' + value + expires + '; path=/'; //Hier wird alles zusammengebaut
09. }
10. function readCookie(name) { /* Analog zur Session! */ }
11. function eraseCookie(name) { createCookie(name, '', -1); } //Löschen mit negativem Ablaufdatum
12. document.getElementById('ex21-name').onchange = function() { //Was passiert beim Wechseln der Selektbox?
13.     document.getElementById('ex21-value').value = readCookie(this.value);//Auslesen mit Wert der Box
14. };
15. document.getElementById('ex21-save').onclick = function() {
16.     createCookie(document.getElementById('ex21-name').value, document.getElementById('ex21-value').value, 7);
17.     document.getElementById('ex21-name').onchange(); //Wechseln (simulieren) zur Wert-anzeige
18. };
19. document.getElementById('ex21-erase').onclick = function() {
20.     eraseCookie(document.getElementById('ex21-name').value); //Aktuell ausgewählten Namen löschen
21.     document.getElementById('ex21-name').onchange();//Wechseln (simulieren) zur Wert-anzeige
22. };
```

WebStorage

- Mit WebStorage haben wir nun ein mächtiges Werkzeug, dass uns nicht nur jede Menge Speicher garantiert (min. **5 MB**), sondern auch nur Clientseitig abläuft
- Um nicht über die Begrenzung hinaus zu schießen hilft folgender Code:

```
01. try { /* try-catch-Block analog zu C++ / C# / ... mit Code um Items zu setzen */ }
02. catch (e) { /* Ein Fehler tritt auf - Fehlerbehandlung */
03.     if (e === QUOTA_EXCEEDED_ERR) { //Wir haben unser Quota (Speicherplatzbegrenzung) gesprengt!
04.         /* ... */ }
```

```
05. }
```

- Das window-Objekt enthält zwei Objekte, localStorage (permanent, sofort über mehrere Fenster hinweg verfügbar) und sessionStorage (flüchtig, nur in dem aktuellen Fenster solange die Seite läuft aktiv)
- Bei den Storage-Elementen kann man im Prinzip nur die folgenden Zugriffen tätigen:

```
01. var enthalteneElemente = localStorage.length; /* Liefert die Anzahl der enthaltenen Elemente zurück */
02. var name = localStorage.key(0); /* Fragt den Schlüsselnamen über den Index ab */
03. var value = localStorage.getItem(name); /* Liest einen Eintrag aus */
04. localStorage.removeItem(name); /* Entfernt einen Eintrag */
05. localStorage.setItem(name, value); /* Erstellt / Ersetzt einen Wert */
06. localStorage.clear(); /* Entfernt alle Einträge */
```

Referenzen:
Codeproject Artikel zu HTML5 [http://www.codeproject.com/Articles/233786/HTML5-The-New-Frontier]
David Walsh über HTML5 Storage [http://davidwalsh.name/html5-storage]
Wikipedia Eintrag [http://en.wikipedia.org/wiki/Web_Storage]
W3C Spezifikation [http://dev.w3.org/html5/webstorage/]
Beispiel zur Verwendung von WebStorage [http://onepixelahead.com/2010/08/23/html5-web-storage-example/]

WebSQL

- Alle Browser unterstützen die Storage-Objekte (IE ab 8, Chrome ab 4, Firefox ab 2, ...):

```
01. if (typeof(localStorage) == 'undefined') //Gibt das localStorage Objekt (oder z.B. sessionStorage) Objekt nicht
02.     alert('Das LocalStorage-Element wird nicht unterstützt - bitte den Browser erneuern...');
```

- Aufgrund der Verbreitung und Sicherheit (Zugriff nur auf Objekte der selben Domäne) eignen sich die Storage Objekte v.a. für clientseitige optimierte Cookies mit Cache-Verhalten (z.B. verwendet von Google-Mail)
- Dieses Konzept sollte daher auch für komplexere Objekte (Datenbanken) übernommen werden
- Leider wurde das ursprüngliche Konzept (WebSQL) nicht gut umgesetzt
- Mittlerweile wurde diese API gestrichen und durch eine neue, IndexedDB ersetzt:

```
01. var request = indexedDB.open("DBName", "Beschreibung"); //Öffnen die Datenbank
02. request.onsuccess = function(event) { //Öffnen erfolgreich - Erstellen eine Datenbank
03.     var db = event.result; //Benutzen die Event-Argumente
```

```
04.     request = db.createObjectStore("TabName", "id", true); //Tabellenname, Schlüssel, Auto-Inkrement setzen
05.     request.onsuccess = function() { db.setVersion("1").onsuccess = function(event) { loadData(db); }; }; };
```

- Vorteil an der neuen API ist die Einfachheit und Ausrichtung auf asynchrone Aufrufe

Referenzen:
Wikipedia Artikel zu IndexedDB [http://en.wikipedia.org/wiki/Indexed_Database_API]
Wikipedia Artikel zu WebSQL [http://en.wikipedia.org/wiki/Web_SQL_Database]
MDN Beispiel zu WebSQL und IndexedDB [http://hacks.mozilla.org/2010/06/comparing-indexeddb-and-webdatabase/]
MDN über IndexedDB [https://developer.mozilla.org/en/IndexedDB]

Beispiel (22)

- Hierbei handelt es sich um das selbe Beispiel wie bei Cookies – aber es gibt einige (große) Unterschiede
- Wichtig: Testen der unterschiedlichen Typen (Lokal und Session) zwischen 2 Fenstern (Tabs)

```
01. /* Hier sind KEINE zusätzlichen Methoden um den Zugriff zu erleichtern ! */
02. //Zugriff auf die sessionStorage ermöglichen
03. var storage = sessionStorage;
04. document.getElementById('ex21-name').onchange = function() { //Beim Wechsel anzeigen
05.     document.getElementById('ex21-value').value = storage.getItem(this.value);
06. };
07. document.getElementById('ex21-save').onclick = function() { //Speichern im aktuellen Element
08.     storage.setItem(document.getElementById('ex21-name').value, document.getElementById('ex21-value').value);
09.     document.getElementById('ex21-name').onchange(); //Wechseln
10. };
11. document.getElementById('ex21-erase').onclick = function() {
12.     storage.removeItem(document.getElementById('ex21-name').value); //Den Eintrag löschen
13.     document.getElementById('ex21-name').onchange(); //Wechseln
14. };
15. document.getElementById('ex22-type').onchange = function() { //Hier wollen wir den Modus festlegen
16.     storage = this.selectedIndex ? window.localStorage : window.sessionStorage; //Entweder local oder session
17.     document.getElementById('ex22-name').onchange(); //Wechseln
18. };
```

Begrenzungen

- Die Webdatenbanken sind im Moment (auch aufgrund der unruhigen Spezifikation) noch weite Zukunftsmusik
- WebStorage bietet wirklich viele Optionen – allerdings sind einige Details zu beachten
- Eine Stolperfalle findet sich mit Firefox Cookie Security:

```
01. var cookiesEnabled = (function() { //Feststellen ob die Cookies verfügbar sind
02.     var id = new Date().getTime(); //Erstellen ein Beispielcookie mit der aktuellen Zeit
03.     document.cookie = '__cookieprobe=' + id + ';path=/'; //Setzen das Cookie (bzw. die Session)
04.     return (document.cookie.indexOf(id) !== -1); //Wenn unser Wert vorhanden ist sind Cookies aktiviert
05. })();
```

  In Firefox funktioniert WebStorage nur bei aktiven Cookies, daher ist dieser Code notwendig!
- Wichtig ist immer die maximale Größe zu beachten – hier hilft der try-catch-Block wie vorhin gelistet
- Eine weitere Einschränkung erhält man durch die Verwendung von Strings
- Wir können außer Strings keine weiteren Daten verwenden – dies kann aber durch **JSON** umgangen werden
- JSON ist fast immer die Lösung für "Wir akzeptieren alle Daten, solange diese Daten Strings sind"-Probleme
- Leider sind die Ereignisse für das Ändern von Storage-Daten bisher nur in Opera implementiert

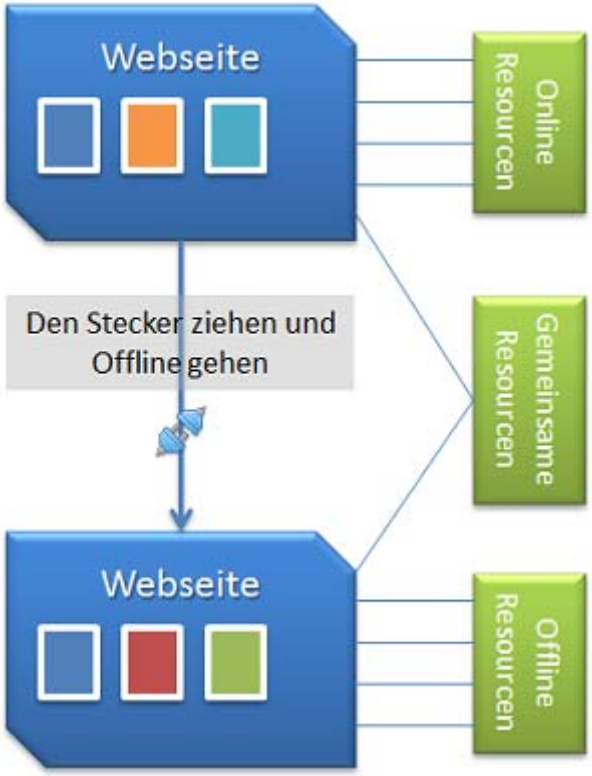
Referenzen:
Einführung in WebStorage [http://sixrevisions.com/html/introduction-web-storage/]
W3C über Web Storage [http://dev.w3.org/html5/webstorage/]
Is 5MB the de facto limit for W3C Web Storage? [http://stackoverflow.com/questions/3232872/is-5mb-the-de-facto-limit-for-w3c-web-storage]
Web Storage: easier, more powerful client-side data storage [http://dev.opera.com/articles/view/web-storage/]

Den Stecker ziehen

- Eine sehr coole Möglichkeit haben wir durch die sog. Manifestdateien erhalten
- Wir können nun genau spezifizieren
  - Was (Welche Dateien bzw. Ordner?)
  - Wann (Online oder Offline?)
  - Unter welchen Bedingungen (Url?)

  geladen werden soll um z.B. eine (eingeschränkte) Offline-Seite bieten zu können
- Die Vorteile liegen auf der Hand: Wir können spezielle Skripten schalten die z.B. Daten speichern (cachen) und bei erneuter Verbindung senden (release)
- Außerdem können wir gezielt das Browser-Cache Verhalten bestimmen

- Wir können dies auch nutzen um den Nutzer eine spezielle - für Offline gebaute - Version der Seite anzubieten
- Alles in allem ermöglicht uns diese Technologie echte Applikationen!



Dank HTML5 kann man dem Browser konkrete Vorgaben machen was (bzw. wie die Seite) im Falle einer aktiven und inaktiven Verbindung angezeigt werden soll

Referenzen:
John Allsopp über das Manifest [http://www.webdirections.org/blog/get-offline/]
IBM zu HTML5 Manifestdateien [http://www.ibm.com/developerworks/web/library/wa-offlineweb/index.html]
Präsentation über HTML Performance [http://www.slideshare.net/souders/high-performance-html5-sf-html5-ug]
News auf Golem zu ManifestR [http://www.golem.de/1107/84910.html]

Das Manifest

- Alles fängt im `<html>`-Tag an:

```
01. <!DOCTYPE html>
02. <html lang="de" manifest="/site.appcache">
03. <!-- Inhalt! -->
```

```
04. </html>
```

- Der `applicationCache` wird nur dann neu geladen, wenn sich die Manifest-Datei geändert hat
- Daher ist es sinnvoll einen Kommentar mit Versionsbeschreibung einzubauen (zur Änderung)
- Wichtig ist, dass der Server die Manifest-Datei mit dem MIME-Type `text/cache-manifest` liefert
- Wir können über `navigator.onLine` rausfinden ob der Browser On- oder Offline ist
- Dies ist allerdings nicht gerade sinnvoll, geschickter ist entsprechende Dateien bei entsprechenden Verbindungsstatus einzubinden
- Jedes Manifest hat 3 Sektionen: Gemeinsame-, Offline- und Online-Resourcen
- Eine Beispielmanifest:

```
01. CACHE MANIFEST
02. # Die oberste Zeile ist notwendig!
03. CACHE:
04. index.html
05. cache.html
06. style.css
07. image1.png
08. # Fallback hier (für Offline)
09. FALLBACK:
10. / fallback.html
11. # Sachen die niemals gecached werden
12. NETWORK:
13. network.html
```

- Die Fallback Sektion gibt an was mit welchem zu ersetzen ist (z.B. `/` (d.h. alles was nicht gefunden wird) mit `fallback.html`

Referenzen:
MDN zum application cache [https://developer.mozilla.org/en/Using_Application_Cache]
HTML5 Demo zum Manifest [http://html5demos.com/offlineapp]
HTML5 Rocks zum Manifest [http://www.html5rocks.com/en/tutorials/appcache/beginner/]
Super Tutorial vom HTML5 Dr. [http://html5doctor.com/go-offline-with-application-cache/]
Ben Nadel über das Cache Manifest [http://www.bennadel.com/blog/1944-Experimenting-With-HTML5-s-Cache-Manifest-For-Offline-Web-Applications.htm]

Beispiel (23)

- Hier die HTML-Datei:

```
<!DOCTYPE html>
<html lang="de" manifest="/manifest.appcache">
<meta charset=utf-8 />
<title>Tolle Offline-Applikation</title>
<script src="jquery.js"></script>
<script src="online.js"></script>
<script src="site.js"></script>
</html>
```

  Nun das zugehörigen Manifest:

```
CACHE MANIFEST
# -- 2011-12-05 -- (v1)
CACHE:
index.html
jquery.js
site.js
FALLBACK:
online.js offline.js
NETWORK:
same.html
```

- Die JavaScript-Datei *site.js*:

```
01. $('<span/>').text(siteStatus).css('font-weight', 'bold').appendTo('body');
02. if(onLine) {
03.     //Was nützliches machen!
04.     $('<div/>').text('Bitte folgende Online-Angebote überprüfen: ').appendTo('body');
05. } else {
06.     //Spezielle Offline Anzeige
07.     $('<div/>').text('Wir haben leider kein Offline-Angebot').appendTo('body');
08. }
```

Die JavaScript-Datei *online.js*:

```
01. var onLine = true;
02. var siteStatus = 'Die Seite ist online!';
```

Die JavaScript-Datei *offline.js*:

```
01. var onLine = false;
02. var siteStatus = 'Die Seite ist offline!';
```

Referenzen:
Das Beispiel [http://html5skript.florian-rappl.de/offlineexample/index.html]

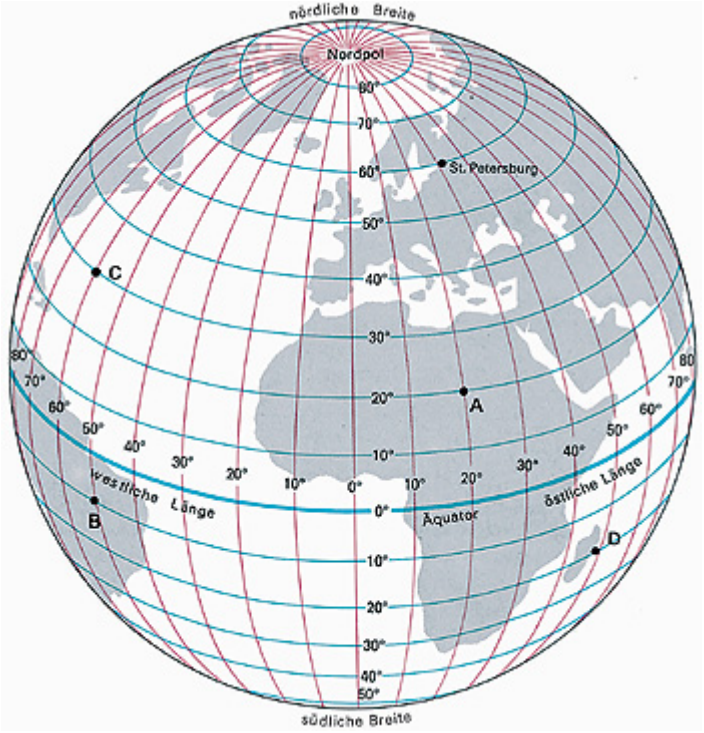
Geolocation

- Im `navigator`-Objekt des `window`-Objektes befindet sich neben allerlei Informationen über den Browser auch noch das `geolocation`-Objekt
- Bei Geolocation-Ereignissen erhalten wir immer ein `position`-Objekt, welches immer `coords` (Koordinaten) und eine `timestamp` (aktuelle Zeit) enthält
- Das Coordinates-Objekt, welches wir sehr häufig verwenden, enthält folgende Eigenschaften:

```
01. var pos = ...geolocation.lastPosition.coords;
02. var x = pos.latitude;//Breitengrad in Grad
03. var y = pos.longitude;//Längengrad in Grad
04. var hor_tol = pos.accuracy; //Genauigkeit in m
05. //Ab hier Angaben die NULL sein können
06. var h = pos.altitude;//Höhe in m
07. var ver_tol = pos.altitudeAccuracy;/* Genauigkeit der Höhenmessung in m */
```

```
08. var dir = pos.heading;/* Richtung (in Grad) mit 0°=Norden (im Uhrzeigersinn) */
09. var v = pos.speed;//Geschwindigkeit in m/s
```

- Da die Suche nach Koordinaten oftmals dauert ist die Geolocation API über Callbacks aufgebaut



Der Breitengrad verläuft von -90° (Südpol) bis 90° (Nordpol) und der Längengrad verläuft über 0° in England (London) bis hin zu 180° im paz. Ozean

Referenzen:
Wikipedia über die geo. Länge [http://de.wikipedia.org/wiki/Geographische_Länge]
Wikipedia über die geo. Breite [http://de.wikipedia.org/wiki/Geographische_Breite]
Codeproject Artikel über HTML5 [http://www.codeproject.com/Articles/233786/HTML5-The-New-Frontier]
Ein Tutorial [http://9elements.com/html5demos/geolocation/]
Ein kleines Demo [http://maxheapsize.com/static/html5geolocationdemo.html]
Die W3C Spezifikation [http://dev.w3.org/geo/api/spec-source.html]

Nützlicher Einsatz

- Mit Geolocation kann man sehr leicht interaktive Services aufbauen, z.B. in den Bereichen
  - Taxis, ÖPNV bzw. Reiseplanung um den aktuellen Standort zu ermitteln
  - Webshops um eine direktere und genauere Ermittlung der Versandkosten durchzuführen

    - Seiten von Unternehmen mit mehr Filialen können auf die Seite des nähesten Standortes weiterleiten
    - Anzeigen können noch regionalen Geschaltet werden
- Jede der Anwendungen läuft auf eine Nutzung der Koordinaten des Benutzers hinaus, um je nach Wert das Angebot zu filtern
- Im Prinzip läuft dies auf folgende Syntax hinaus:

```
01. $(document).ready(function() { /* Verwenden der jQuery Ready Methode für JavaScript */
02.     // Grundansicht der Seite laden (ohne Geo-Daten)
03.     navigator.geolocation.getCurrentPosition(GeolocationPosition); /* analog dazu: watchPosition() */
04. });
05. function GeolocationPosition(position) { /* Callback wenn die Daten ermittelt worden sind */
06.     var lat = position.coords.latitude;
07.     var lon = position.coords.longitude;
08.     //Weitere Inhalte laden (speziell für diese Koordianten!)
09. }
```

Referenzen:
Beispiel zu Geolocation mit HTML5 [http://maxheapsize.com/2009/04/11/getting-the-browsers-geolocation-with-html-5/]
Kleines Demo [http://html5demos.com/geo]
10 Gründe wieso Geolocation die Welt verändert [http://nowsourcing.com/2010/07/27/10-ways-geolocation-is-changing-the-world/]
12 coole Ideen für Geolocation auf MSDN [http://msdn.microsoft.com/en-us/magazine/hh563893.aspx]

Beispiel (24)

```
01. if (navigator.geolocation) {
02.     var options = { enableHighAccuracy: false, /* Akzeptieren alles */
03.                     timeout: 5000, /* Zeit in ms für TimeOut - 0 bedeutet kein TimeOut */
04.                     maximumAge: 0 /* 0 = Muss neu, Infinity = Muss alt, ansonsten in ms */
05.     };
06.     navigator.geolocation.getCurrentPosition(geoSuccess, geoFailure, options);
07. } else geoFailure(); // Wenn kein geolocation-Objekt existiert ist der Browser zu alt
08. function showLocation(lat, lng) { //Dient zum Anzeigen der Position
09.     /* Google Maps Code (Anzeigen der Karte) */ }
10. function geoSuccess(position) { //Wird im Falle einer erfolgreichen Geopositionsabfrage aufgerufen
11.     document.getElementById('ex24-loc').value = position.coords.latitude + "," + position.coords.longitude;
```

```
12.     showLocation(position.coords.latitude, position.coords.longitude); //Anzeige der Position
13. } //Ansonsten wird geoFailure aufgerufen
14. function geoFailure() { document.getElementById('ex24-loc').value = 'Nicht verfügbar...'; }
```

Responsive Design

- Mit CSS3 wurde auch ein neuer Ansatz für Webseiten design eingeführt: sog. Media Queries
- Dies bedeutet, dass die Webseite auf das verwendete Gerät (z.B. Mobiltelefon, PC, ...) und seine Eigenschaften (z.B. Größe, Auflösung, ...) reagiert
- Aus CSS 2.1 kennt man schon das `media` Attribut von Stylesheets, z.B.

```
01. <link rel="stylesheet" media="print" href="druck.css" />
02. <link rel="stylesheet" media="screen" href="layout.css" />
```

  Nun wurde in CSS3 das Attribut um weitere Möglichkeiten erweitert, z.B.

```
01. <link rel="stylesheet" href="screenklein.css" media="screen and (max-width: 600px)" />
02. <link rel="stylesheet" href="allesklein.css" media="all and (max-width: 600px)" />
03. <link rel="stylesheet" href="smartphone.css" media="(max-device-width: 600px)" />
```

- Neben and (Verknüpfen von Bedingungen) gibt es noch das Komma, welches als ODER eingesetzt wird
- Weitere Abrufbare Eigenschaften sind (Auszug): `height` bzw. `device-height`, `orientation` (portrait, landscape), `aspect-ratio` bzw. `device-aspect-ratio`, `color` (Farbtiefe in Bits) und `resolution` (Dichte der Pixel in dpi)

Referenzen:
Responsive Design anschaulich erklärt [http://www.thismanslife.co.uk/projects/lab/responsiveillustration/]
Artikel bei Heise über Media Queries [http://www.heise.de/ix/artikel/Allen-recht-1058764.html]
Die W3C Spezifikation [http://www.w3.org/TR/css3-mediaqueries/]
CSS Tricks zum Thema [http://css-tricks.com/6731-css-media-queries/]
DrWeb Artikel [http://www.drweb.de/magazin/media-queries-mobile-version-von-websites-mit-css3-erstellen/]

Die @-Regeln

- Wir kennen bereits die @-Regeln `@fontface` und `@keyframes`
- Durch `@charset` kann man die Kodierung festlegen (UTF-8 ist Standard!), z.B.

```
01. @charset "ISO-8859-15";
```

- Über `@import` können wir weitere (externe) Stylesheets einbinden, z.B.

```
@import url("/css/main.css"); /* Oder mit Media-Query */
@import url("style.css") screen and (max-width: 600px);
```

- Dank `@media` haben wir Zugriff auf die Media-Queries, z.B.

```
@media print { /* Selektoren mit Regeln */ }
@media screen and (max-width: 600px) { /* Selektoren mit Regeln */ }
```

- Mit @page können wir die Druckausgabe anpassen (mehr dazu später) und mit @namespace erlaubt uns Namensräume (für Elemente) einzustellen, z.B.

```
@namespace foo "http://example.com/ns/foo";
foo|#mydiv { /* Regeln */ } /* Für dieses Element mit diesem xmlns Attribut */
```

Referenzen:
W3C Spezifikation der Mediatypen [http://www.w3.org/TR/CSS2/media.html]
HTMLDog zu den @-Regeln [http://www.htmldog.com/guides/cssadvanced/atrules/]
Referenz auf Sitepoint [http://reference.sitepoint.com/css/atrulesref]
CSS Tricks dazu [http://css-tricks.com/6731-css-media-queries/]

Die @media-Regel

- `@charset` und `@import` müssen immer vor allen Regeln festgelegt werden - um die in `@fontface` festgelegte Schrift nutzen zu können, muss diese ebenfalls vorab festgelegt sein
- Analoges gilt für die @namespace Deklaration - auch hier muss die Regel vor der Verwendung festgelegt sein
- Bei `@media` kann man folgende Gerätearten direkt aufrufen: `all` (Alles), `braille` und `embossed` (Blindenschrift), `handheld` bzw. `tty`, `print`, `projection`, speech (Sprachausgabe) bzw. (veraltet) `aural`, sowie `tv` und `screen`
- Wie kann dergleichen aussehen? Wir schreiben uns CSS-Dokument und bauen dies z.B. so auf:

```
/*Gemeinsame Regeln */
@media screen /* Nur für Screen */
{
    body { width: 75%; }
}
@media print /* Nur für Print */
{
```

```
    body { width: 100%; }
}
```

- Die @media-Regeln können auch sehr speziell und kompliziert aufgebaut werden (AND, OR, ...)

Referenzen:
Sitepoint Referenz zu @media [http://reference.sitepoint.com/css/at-media]
CSS Tricks Artikel [http://css-tricks.com/6731-css-media-queries/]
Mediaevent über die @media-Regel [http://www.mediaevent.de/css/media-type.html]
SelfHTML zum Media Attribut [http://de.selfhtml.org/css/formate/einbinden.htm]

Beispiel (25)

```
body > div { /* Allgemeine Angaben */
    position: absolute;
    left: 50%; top: 50%;
    border: 1px solid #AAA;
    /* Nun spezielle Angaben - erstmal für unseren Bildschirm */
    width: 600px; height: 400px;
    margin: -221px 0 0 -321px; padding: 20px;
} /* Nun für Geräte mit einer maximalen Breite von 700px und einer minimalen Breite von 500px */
@media all and (max-width: 700px) and (min-width: 500px) {
    body { font-size: 10px; } /* Schriftart neu setzen etc. */
    body > div {
        width: 400px; height: 300px;
        margin: -161px 0 0 -211px; padding: 10px;
    }
} /* Und abschließend noch für sehr kleine Geräte (<= 500px Breite) */
@media all and (max-width: 500px) {
    body { font-size: 8px; } /* Schriftart neu setzen etc. */
    body > div {
        width: 200px; height: 150px;
        margin: -81px 0 0 -106px; padding: 5px;
    }
    label { margin: 0; } /* Labels hätten zu viel Abstand für diese Auflösung */
```

```
23. }
```

Referenzen:
Das Beispiel [http://html5skript.florian-rappl.de/uiexample/mediaquery.html]

Reactive Design

- Im Prinzip handelt es sich bei dem Responsive Design um ein Reactive Design (es wird nicht geantwortet, d.h. auf den Benutzer eingegangen, sondern auf die Umgebung reagiert)
- Designs sollten so aufgebaut werden, dass diese sehr schnell für verschiedene Plattformen portiert werden können
- Dies schließt relative Schriften mit ein - wenn man jedem Element eine neue Schriftgröße zuweist sind mehr Änderungen notwendig als bei Zuweisung von Prozentangaben relativ zur im body-Selektor eingestellten Schriftgröße
- Zusätzlich zu den `@media`-Regeln sollte man noch weitere (HTML) Spezifikationen beachten
- So ist es essentiell zu den **Viewport** für Smartphones einzustellen
- Die Einstellung des Viewports geschieht im <head>-Tag z.B. mit

  ```
  01. <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no" />
  ```

  Legt in diesem Fall fest, dass die Seite die volle Breite hat und dies 100% Zoom (deaktiviert) entspricht
- Bilder sollten mit `img { max-width: 100%; }` behandelt werden um bei Smartphones nicht zu stören
- Für Smartphones gibt es noch jede Menge weiterer Tricks (Mobile First statt Desktop First) inklusive einiger besonderer CSS-Regeln wie z.B. `-webkit-tap-highlight-color: #FFF`

Referenzen:
Nützliche Media Queries von Andy Clarke [http://www.stuffandnonsense.co.uk/blog/about/hardboiled_css3_media_queries/]
W3C Spezifikation zu Geräteadaptionen [http://dev.w3.org/csswg/css-device-adapt/]
Testseite für Verschiedene Eigenschaften [http://www.quirksmode.org/m/tests/widthtest.html]
Sammlung von Beispielen zu Media Queries [http://www.theinspirationblog.net/showcases/css3-media-queries-in-the-wild/]
Artikel über Reactive Design [http://netuncovered.com/2011/05/reactive-web-design/]

Professioneller Druck

- Mit den @page-Anweisung kann man wie bei den Media-Queries spezielle Regeln definieren
- Wir auch können zwischen ungeraden (*left*) und geraden (*right*) Seiten unterscheiden, z.B.

  ```
  01. @page { /* Allgemeine Regeln für Druckseiten */ }
  02. @page : left { /* Spezielle Regeln für ungerade Seiten */ }
  03. @page : right { /* Regeln für gerade Seiten */ }
  ```

```
04. @page : first { /* Regeln für die erste Seite */ }
```

- Ein spezielles Design reicht oft nicht aus um Webseiten zum Drucken zu optimieren, so müssen wir den Content auch noch beachten, z.B. stellt sich als Frage was man mit Links macht
- Eine Möglichkeit besteht darin die Referenzen auf die Links verweisen anzuzeigen, z.B. über

```
01. a:after { content: '(' attr(href) ')'; /* ... */ } /* Betrifft alle Links */
02. a[href^="http://"]:after { /* ... */ } /* nur für externe Links mit http */
03. a[href^="https://"]:after { /* ... */ } /* nur für externe Links mit https */
04. a[href^="#"] { display: none; } /* Verhindert die Anzeige interner Links */
```

  Die letzte Anweisung stellt eine spezielle Behandlung von internen (Hash) Links dar

- Man sollte außerdem immer auf weitere Regeln wie z.B. `page-break-before: always` achten - hiermit kann man sehr leicht kontrollieren was zusammen bleiben soll

Referenzen:
Artikel im Smashing Magazine [http://coding.smashingmagazine.com/2011/11/24/how-to-set-up-a-print-style-sheet/]
SelfHTML zu diesem Thema [http://aktuell.de.selfhtml.org/artikel/css/drucklayout/]
Eric Meyer über Printstylesheets [http://www.alistapart.com/articles/goingtoprint/]

Asynchroner JavaScript

- JavaScript war schon immer ein Single-Threaded Problem, d.h. wir hatten keine Chance eine Operation abzubrechen
- Mit den WebWorker haben wir nun die Möglichkeit Multithreading zu betreiben
- Zunächst sollte man dazu überprüfen ob der Browser dies unterstützt

```
01. if(typeof Worker != "undefined") {
02.     /* Wir können den WebWorker anwenden! */
03. }
```

- Wie schon aus echten Programmiersprachen bekannt dürfen wir keine Cross-Threading-Zugriffe tätigen
- Daher läuft die Kommunikation über die Messaging API ab - ein Beispiel

```
01. var worker = new Worker('my_worker.js');
02. worker.postMessage('Hallo!');
03. worker.onmessage = function(event) {
04.     console.log(event);
```

```
05. };
```

- Der Worker hat keinen Zugriff auf `window` oder den DOM - sein Basisobjekt ist das `Worker`-Objekt

Referenzen:
Wikipedia Artikel zu WebWorker [http://en.wikipedia.org/wiki/Web_Workers]
Worker bei WHATWG [http://www.whatwg.org/specs/web-apps/current-work/multipage/workers.html]
John Resig über WebWorker [http://ejohn.org/blog/web-workers/]
MDN Artikel zum Thema [https://developer.mozilla.org/en/Using_web_workers]
Tutorial im Internetmagazin [http://www.internet-magazin.de/ratgeber/javascript-parallel-ausfuehren-1107308.html]
Artikel bei Codeproject [http://www.codeproject.com/KB/solution-center/HTML5-Web-Workers.aspx]

Warum WebWorker?

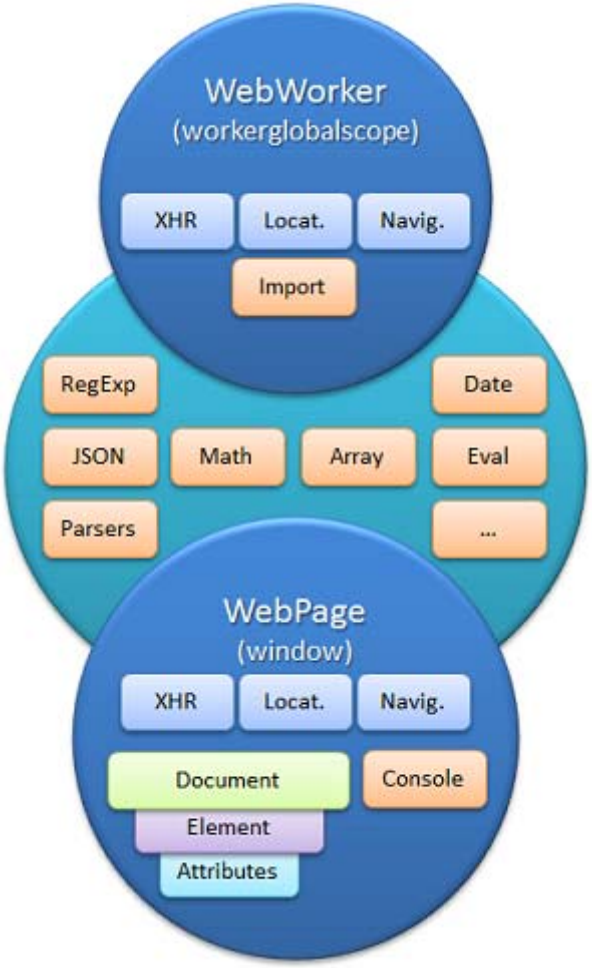
- Die Applikation soll immer responsive bleiben (immer reagieren können)
- Als Faustregel gilt: Wenn eine Operation mehr als **50 ms** brauchen kann, sollte diese asynchron sein
- Ein WebWorker kann von außen über `terminate()` abgebrochen werden
- Innerhalb eines Workers haben wir nur die Möglichkeiten über `postMessage()` an die Quelle zu senden oder diese über das `onmessage`-Ereignis zu empfangen
- Diese Möglichkeiten unterliegen der Messaging API und können nur Strings senden und empfangen (JSON!)
- Des Weiteren können wir AJAX Requests, WebSockets, alle JS-Methoden wie `eval()`, `isNaN()` etc. und alle Timer wie `setInterval()` verwenden
- Mit Hilfe von `importScripts` kann man weitere Skripte (z.B. jQuery) reinladen
- Das `location`-Objekt enthält die Adresse des WebWorkers
- Worker dürfen das `navigator`-Objekt verwenden: `appName`, `appVersion`, `platform` und `userAgent`
- Über `close()` können wir den Worker innerhalb schließen
- Innerhalb des Workers ist `this` gleich der Worker-Instanz

Referenzen:
WebWorker bei Sitepoint [http://www.sitepoint.com/javascript-threading-html5-web-workers/]
WebWorker Basics [http://weblog.bocoup.com/javascript-web-workers-from-basics-to-jquery-hive-part-ii-browser-implementations]
WebWorker mit JavaScript [http://thoriumware.com/?p=118]
IE Blog auf MSDN zu WebWorker [http://blogs.msdn.com/b/ie/archive/2011/07/01/web-workers-in-ie10-background-javascript-makes-web-apps-faster.aspx]

WebWorker debuggen

- Mit WebWorkern zielt man noch mehr in die Richtung *unobtrusive JavaScript*
- Man möchte also vermeiden, dass JavaScript durch eine ständige Berechnung die Seite lahmlegt

- Neben dem DOM fehlt einem Worker auch der Zugriff auf die Konsole
- Dies erschwert das Debugging ungemein
- Über Haltepunkte und dergleichen ist Debugging dennoch möglich
- Es ist aber möglich `console.log()` über den `MessageChannel` zu simulieren[1]
- Dies funktioniert in IE10, Opera, Chrome, Safari – jedoch (noch) nicht in Firefox
- Nach Einbinden der ConsoleWorker.js-Datei ist `console.log()` auch in der Konsole verfügbar
- Tipp zum Testen: zunächst ohne WebWorker versuchen und wenn der Code ok ist auf WebWorker umstellen



Trotz einiger Gemeinsamkeiten fehlen dem WebWorker viele wichtige Objekte wie z.B. der Zugriff auf die Konsole

Referenzen:

Codeproject Artikel über WebWorker [http://www.codeproject.com/KB/solution-center/HTML5-Web-Workers.aspx]
[1]Anleitung zur Verwendung der Konsole im Worker [http://www.davidflanagan.com/2011/01/consolelog-for.html]
Mehr über den MessageChannel [http://www.w3.org/TR/webmessaging/#messagechannel]
Das Fountain WebWorker Demo [http://ie.microsoft.com/testdrive/Graphics/WorkerFountains/Default.html]
Code der WorkerConsole auf GitHub [https://github.com/davidflanagan/WorkerConsole]
MSDN Developer Guide zu WebWorker [http://msdn.microsoft.com/library/hh673546.asp]
Unaufdringlicher JavaScript [http://en.wikipedia.org/wiki/Unobtrusive_JavaScript]

Beispiel (26)

```
01. //---JavaScript dieser Datei---
02. if(typeof Worker != "undefined") {
03.     var w = new Worker('prime.js');
04.     w.onmessage = function(event) {
05.         console.log(event); //Wir wollen das ganze auch als Event-Log sehen!
06.         var el = document.createElement('span'); //Fügen dem DOM ein span Element hinzu
07.         el.innerHTML = event.data + ", ";
08.         document.getElementById('ex26-box').appendChild(el);
09.     };
10.     w.postMessage(JSON.stringify({ start : 100, end : 10000})); // Starten WebWorker mit Argumenten
11. }
12. //---JavaScript der Datei "prime.js"---
13. /* Was passiert wenn wir eine Nachricht erhalten?! */
14. onmessage = function(event) {
15.     var obj = JSON.parse(event.data); /* Lesen die Argumente aus (JSON) und */
16.     run(obj.start, obj.end); /* Starten Berechnung von einer Zahl bis zu einer anderen */
17. };
18. function run(s, e) {
19.     /* Primzahlberechnung und im Falle einer Primzahl: */
20.     postMessage(n.toString());
21. }
```

Eigene jQuery Plugins

- Eigene jQuery Plugins zu schreiben ist in vielen Fällen vorteilhaft

- Zum einen modularisiert man seine Arbeit (Wiederverwendbarkeit!), zum anderen schafft man auch mehr Übersichtlichkeit und weniger Fehler
- Code explizit als jQuery Plugin zu bauen hat außerdem Vorteile für die tägliche Verwendung
- Am Ende möchte man bequeme jQuery Methoden à la `$(...).meinPlugin(...)` schaffen
- In JavaScript können wir dies über `prototype` erreichen - dies wurde in jQuery abgekürzt:

```
01. (function($) { // Anonyme Methode erstellen, mit einem Argument ($)
02.   $.fn.myPluginName = function() { //Legen unsere Methode fest - statt prototype einfach fn schreiben
03.     //Hier kommt der Plugin Code rein!
04.   };
05. })(jQuery); //Methode wird sofort mit dem jQuery Objekt aufgerufen
```

- Das tolle an unserem Plugin: Es wird von jQuery aufgerufen und besitzt mit `this` einen Zeiger auf die aktuelle jQuery Auswahl, z.B. ist bei `$('.klasse').myPluginName()` der `this` Zeiger äquivalent zu `$('.klasse')`
- Am Ende **sollten** wir auch dieses zurückgeben - dies ist nur notwendig wenn die Rückgabe ungleich `return this` wäre

Referenzen:
Tutorial für ein eigenes Plugin [http://blog.jeremymartin.name/2008/02/building-your-first-jquery-plugin-that.html]
Designmuster für gute Plugins [http://coding.smashingmagazine.com/2011/10/11/essential-jquery-plugin-patterns/]
jQuery Deconstructed [http://www.keyframesandcode.com/resources/javascript/deconstructed/jquery/]
Infos von der jQuery Seite [http://docs.jquery.com/Plugins/Authoring]
Codeproject Artikel mit einem guten Beispiel [http://www.codeproject.com/KB/scripting/HowToWritePluginInJQuery.aspx]

An Designmuster halten

- Damit andere Leute auch in den Genuss der Plugins kommen können oder um die eigenen Plugins effektiver zu schreiben sollte man sich an einige Regeln bzw. Vorgehensweisen halten
- Ein guter Start für jedes Plugin sieht folgendermaßen aus:

```
01. //Der Strichpunkt dient zum sicheren Abschließen von vorherigen Code
02. ;(function($, undefined) {
03.     var defaults = {
04.         /* Standardoptionen festlegen */
05.     };
06.
07.     $.fn.myPlugin = function(options) {
08.         //Extend verwenden um nicht gesetzte Optionen auszufüllen
09.         var options = $.extend(defaults, options);
```

```
10.
11.         return this.each(function () { //Durch alle Elemente durchgehen und die Veränderung durchführen
12.             /* Was mit $(this) machen -- unser Plugin eben! */
13.         });
14.     };
15. })(jQuery);//da wir nur 1 Argument übergeben aber 2 erwarten wird undefined undefined sein ;-)
```

- Man sollte private Variablen immer privat halten und am Besen eigene Namensräume verwenden
- Es gibt noch mehr Möglichkeiten (Events, Erweiterung bestehender Methoden, ...)

Referenzen:
Regeln für jQuery Plugins [http://coding.smashingmagazine.com/2011/10/11/essential-jquery-plugin-patterns/]
jQuery Plugin Pattern von Adam Novsky [http://milan.adamovsky.com/2010/09/jquery-plugin-pattern-20.html]
Ein einfaches Vorgehensmuster [http://scriptble.com/2011/02/expanding-the-jquery-plugin-development-pattern/]
jQuery Plugins Seite mit jeder Menge Tutorials [http://www.jqueryplugins.com/]

Ein Beispiel: Show-More

```
01. ;(function($, undefined){ /* Dies stellt sicher, dass wir $ verwenden können */
02.     $.fn.truncate = function(options) { /* Benutzer kann eigene Optionen übergeben */
03.         var options = $.extend({ length: 300, moreText: "mehr", lessText: "weniger" }, options); /* Mischen */
04.         return this.each(function() {  /* Die übergebenen Elemente bearbeiten */
05.             var body = $(this).html();  // Uns interessiert der HTML-Inhalt
06.             if(body.length > options.length) { // Und zwar nur dann wenn es zuviel ist...
07.                 var splitLocation = body.indexOf(' ', options.length); //1. Leerzeichen nach Zeichenlänge finden
08.                 if(splitLocation !== -1) { //Wenn wir überhaupt Leerzeichen drin haben
09.                     /* Optionalen Inhalt rausfinden und verstecken */
10.                     moreLink.click(function() { /* Wenn auf Link geklickt wird */
11.                         if(moreLink.text() === options.moreText) { /* Mehr anzeigen */
12.                             moreContent.show('normal');   //Mit jQuery Effekt anzeigen
13.                             moreLink.text(options.lessText); //Linktext verstecken
14.                             ellipsis.css("display", "none"); //Weg mit "..."
15.                         } else {  /* Weniger anzeigen */
16.                             /* Selbes nur umgedreht */
17.                         }
```

```
18.                         return false; //Verhindern dass Link noch irgendwas macht!
19. })}}})}})(jQuery);
20. $('#exmore-box').truncate({ length: 50 });
```

Noch einmal CSS3

- In CSS3 kann man mehrere Hintergründe einbinden, z.B. über

```
01. selektor { background: url('hintergrund1.gif') no-repeat 0 0, url('hintergrund2.gif') no-repeat 100% 0; }
```

- Als Workaround für ältere Browser empfiehlt sich folgendes

```
01. selektor { /* FALLBACK für NON-CSS3 Browser ... */ }
02. body:last-child selektor { /* Für CSS3 Browser */ }
```

- Neben Rahmen über `border` sind auch sog. `Outlines` möglich, z.B.

```
01. selektor {
02. border: 1px solid #000;
03. outline: 1px solid #699;
04. outline-offset: -9px; }
```

- In CSS3 kann man nun auch explizit das Box-Model durch das IE(5)-Model ersetzen:

```
01. selektor { box-sizing: content-box; /* Reguläres Box-Model - *meistens* Standard */ }
02. selektor {     box-sizing: border-box; /* Nicht standardisiertes Model - vorsicht Webkit-Textboxen! */ }
```

Referenzen:
Codeproject Artikel über CSS3 [http://www.codeproject.com/KB/HTML/CSS3.aspx]
Nico Brünjes über Multiple Backgrounds [http://codecandies.de/2009/11/22/kurz-gecodet-multiple-backgrounds/]
CSS Outlines [http://www.css3.info/preview/outline/]
W3C über Border und Content Box [http://www.w3.org/TR/css3-ui/]

JavaScript Getter

- Getter und Setter stellen eine Kapselung von Variablen dar, wobei der Zugriff auf die Variablen wie mit Eigenschaften (d.h. wie öffentliche Variablen) erfolgt

- Nachdem die Variable jedoch gekapselt ist, kann man **VOR** der Zuweisung eine Überprüfung durchführen
- Außerdem ist es möglich Eigenschaften einzuführen, die keine echten Variablen darstellen, aber wie solche angesprochen werden sollen
- Wir können Getter direkt in der Objektdeklaration einbauen, wie z.B.

```
01. var lost = {
02.     loc : "Island",
03.     get location () {
04.         return this.loc;
05.     }
06. };
07. alert(lost.location);
```

- Genauso können bestehende Elemente wie `HTMLElement` erweitert werden, z.B.

```
01. HTMLElement.prototype.__defineGetter__("description", function () { return this.desc; });
02. alert(document.body.description);
```

Referenzen:
MDN Artikel über Get/Set [https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Object/defineProperty]
John Resig über Getter und Setter [http://ejohn.org/blog/javascript-getters-and-setters/]
Robert Nyman über Get/Set [http://robertnyman.com/2009/05/28/getters-and-setters-with-javascript-code-samples-and-demos/]

JavaScript Setter

- Variablen mit Getter zu kapseln ist noch nicht genug - wirklich sinnvoll ist der Setter
- So kann man gezielt auf Variablenänderungen reagieren, z.B. ist folgendes möglich:

```
01. var lost = {
02.     loc : "Island",
03.     set location(val) {
04.         this.loc = val;
05.         alert('The Location has moved! New Location: ' + this.loc);
06.     }
07. };
08. lost.location = "Another island";
```

- Analog zu den Gettern können wir wieder bestehende Objekte erweitern

```
01. HTMLElement.prototype.__defineSetter__("description", function (val) { this.desc = val; });
02. document.body.description = "Beautiful body";
```

- Analog zu den vorherigen Möglichkeiten kann man noch den Weg des IE (ab Version 8) gehen

```
01. Object.defineProperty(obj, name, { get : function() { }, set : function() {} })
```

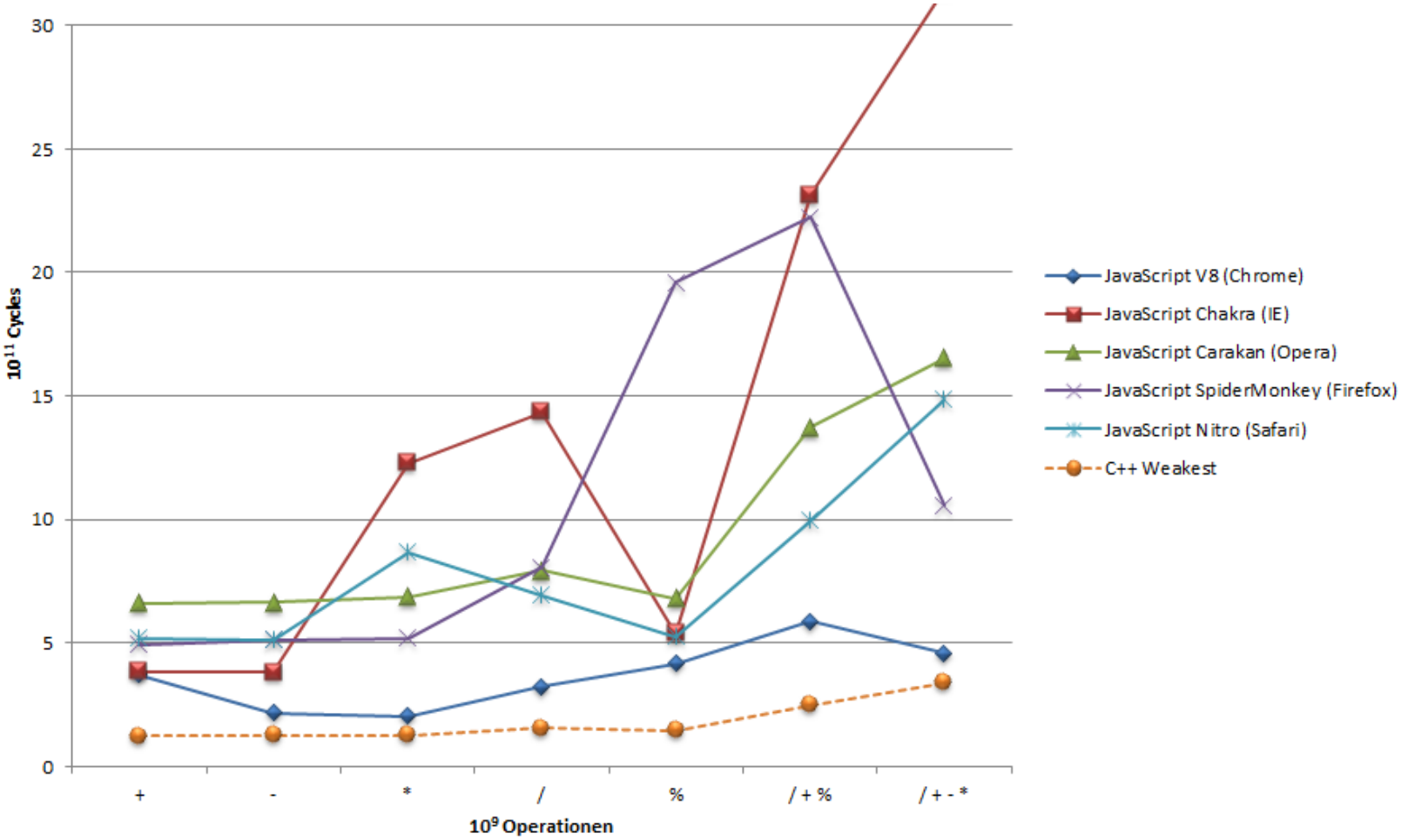
Referenzen:
MDN Artikel über Get/Set [https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Object/defineProperty]
John Resig über Getter und Setter [http://ejohn.org/blog/javascript-getters-and-setters/]
Kurzer Überblick über JS inkl. Get/Set [http://www.html-world.de/program/js_2.php]
Robert Nyman Kompatibilitätsliste [http://robertnyman.com/javascript/#javascript-getters-setters-object-defineproperty-compatibility]

Beispiel (27)

```
01. var o = {}; // Erstellt ein neues Objekt
02. // Verwendung von defineProperty um Get/Set (Accessors) zu setzen
03. Object.defineProperty(o, "a", {
04.             get : function() { return aValue; },
05.             set : function(value) { aValue = value + 5; alert(value); },
06.             enumerable : true,
07.             configurable : true
08.         }
09.     );
10. o.a = 10;
11. alert(o.a);
12. // Beispiel für eine Read-Only Variable mit festen Startwert (Data-Descriptors)
13. var bValue;
14. Object.defineProperty(o, "b", {
15.             value : 'Fester Wert!',    //Startwert (geht nicht bei get/set)
16.             writable : false,        //Schreibbar (geht nicht bei get/set)
17.             enumerable : false,        //Wahr wenn die Eigenschaft im Objekt aufgelistet werden soll
18.             configurable : true        //Wahr wenn die Beschreibung verändert / gelöscht werden darf
```

```
        }
    );
alert(o.b);
o.b = 'Das ist ein Test!'; // Wird wohl nicht gehen - ist Read-Only!
alert(o.b);
```

Die Performance



Performance von JS Engines in Cycles pro Operation (Stand Januar 2012)

Referenzen:

Artikel auf Codeproject [http://www.codeproject.com/KB/tips/Performances.aspx]

Genauigkeit von JavaScript Timern [http://ejohn.org/blog/accuracy-of-javascript-time/]

Performance Tricks

- Um JavaScript schneller laufen zu lassen sollten auf jeden Fall ständige DOM Abfragen vermieden werden
- Bei bekannten Typen empfiehlt es sich === und !== anstelle von == und != zu verwenden (spart Cast!)
- Außerdem sollten verknüpfte Bedingungen so umgestellt werden, dass die einfachsten am Anfang stehen

```
01. var x = 10, y = true;
02. if (y || x * x > 1000) // Viel besser als x * x > 1000 || y
03.     alert('true!');
```

- Die Performance von JavaScript skaliert invers mit der Anzahl der Operationen – weniger ist besser
- Daher gilt immer: Unnötige Operationen weglassen – am besten zusammenfassen wie hier:

```
01. oElement.style.position = 'absolute'; //Schlechter Stil - lauter einzelne Operationen
02. oElement.style.top = '0px'; /* etc... */
03. oElement.setAttribute('style', 'position: absolute; top: 0px; /* etc... */'); //BESSER!
```

Auch bei `for`-Schleifen kann man tricksen, so ist eine der effizientesten Arten folgende

```
01. var anchors = document.getElementsByTagName('a');
02. for (var i = anchors.length; i--; ) { /* Anweisungen hier */ }
```

Referenzen:
Artikel auf Codeproject [http://www.codeproject.com/KB/tips/Performances.aspx]
Schnellere Bedingungen [http://www.sitepoint.com/faster-javascript-condition-expressions/]
Ausführliche Benchmarks von Steve Souders [http://oreilly.com/server-administration/excerpts/even-faster-websites/writing-efficient-javascript.html]
Jede Menge Do's and Dont's von UserJS [http://userjs.org/help/tutorials/efficient-code]
Effiziente for-Schleifen in JavaScript [http://www.impressivewebs.com/javascript-for-loop/]

Das Netzwerk optimieren

- Das Netzwerk wird im Prinzip durch das OSI (Open Systems Interconnection) Modell in 7 Schichten aufgeteilt
- Als Webseitenentwickler interessiert uns allerdings nur die höchste Schicht auf der http als ein auf TCP/IP basierter Service läuft
- Von unseren Webserver interessieren uns Netzwerktechnisch 2 Sachen: Datendurchsatz und Antwortzeit
- Es ist auch über JS möglich diese Werten zu messen

- Die Frage ist nun - wie kann man das Netzwerk aufgrund der gefundenen Daten optimieren?
- Ohne auf den Server direkt zuzugreifen können wir die Resourcen der Seite auf verschiedene Server (sog. Content Delivery Networks, CDN) verteilen und http-Requests generell verringern
- Daher stellen wir z.B. auch jQuery (und andere statische Resourcen) auf einen externen Server
- Am Server können wir DNS-Lookups reduzieren und Redirects jeder Art vermeiden
- Um DNS-Lookups zu reduzieren sollten hohe TTL (Time-To-Live) Werte verwendet werden (ca. 1 Tag)
- Die Browser führen im Regelfall einen eigenen DNS-Cache, welcher häufig TTL Werte ignoriert und eigene Regeln hat

Referenzen:
Artikel im Smashing Magazine [http://coding.smashingmagazine.com/2011/11/14/analyzing-network-characteristics-using-javascript-and-the-dom-part-1/]
Greg's Cable Map [http://cablemap.info/]
Boomerang Projekt für Server Benchmark [https://github.com/bluesmoon/boomerang]
Interessantes RFC mit Details zum Format von DNS Einträgen [http://tools.ietf.org/html/rfc1537]

JS und CSS Tricks

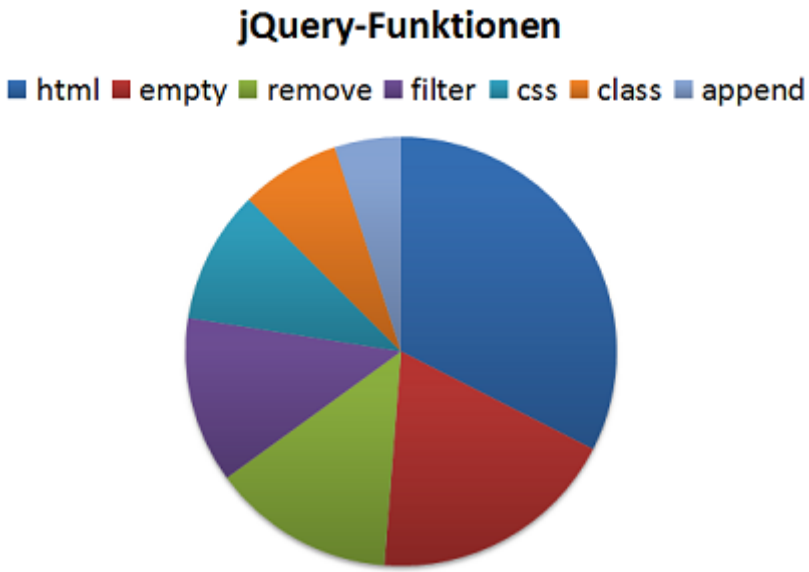
- Während des Downloads eines Skripts im <head>-Tag wird kein weiterer Download gestartet
- Dies ist unabhängig von der Art des Downloads (weiterer Skript, Bild, ...) und daher sehr schlecht für die Seite
- Daher gilt absolut immer: Skripten gehören an das Ende des <body>-Tags!
- Bei Stylesheets ist die Platzierung im <head>-Tag lohnenswert, da ansonsten Rendering verhindert wird
- Daher gilt hier: Stylesheets gehören immer in den <head>-Tag um Rendering zu ermöglichen!
- JavaScript-Dateien sollten immer minifiziert werden (Kommentare, Einrückungen, Leerzeilen und unnötige Auslassungen entfernen)
- CSS-Dateien sollten auch minifiziert werden - wichtig ist, dass alle min. Versionen nur produktiv eingesetzt werden
- Bei JavaScript kann man zusätzlich noch Variablennamen verändern (auf 1-3 Zeichen kürzen)
- Hierbei ist die Wahl des Tools jedoch sehr wichtig, da manche Tools bei manchen Skripten für Fehler sorgen können

Referenzen:
Yahoo Best Practices [http://developer.yahoo.com/performance/rules.html]
Steve Souders über JS/CSS [http://developer.yahoo.com/blogs/ydn/posts/2007/07/high_performanc_5/]
StackOverflow zu JS am Ende der Seite [http://stackoverflow.com/questions/1638670/javascript-at-bottom-top-of-web-page]
JS/CSS minifizieren [http://wolf-u.li/1964/minify-minimize-javascript-js-und-cascading-stylesheets-css-online-durchfuehren/]
JSMIN - Das Original von Douglas Crockford [http://www.crockford.com/javascript/jsmin.html]

Häufige jQuery Befehle

- Schaut man sich eine statistische Erhebung der am meisten verwendeten jQuery Methoden an stellt man fest, dass `ajax()` und `animate()` gar nicht in der Top-Liste vertreten sind

- Statt dessen sind HTML-Helfer wie `html()`, `text()` und `val()` ganz oben
- `empty()` (entfernt alle Unterelemente des aktuellen Selektors) und `remove()` mit `replace()` und `replaceWith()` folgen
- Danach sind Filter-Methoden (`find()`, `eq()`, ...)



Eine Statistik zu den am häufigsten verwendeten jQuery Methoden

- Erst jetzt kommen Veränderungen an den (CSS-)Eigenschaften mit `css()` oder `attr()`
- Auch `addClass()`, `removeClass()` und `toggleClass()` zielen direkt auf CSS ab
- Am Ende der Top-Liste stehen DOM-Manipulationen über `append()` bzw. `appendTo()` oder auch `prepend()` bzw. `prependTo()`
- Erstere fügen Elemente ans Ende der Liste hinzu, letztere fügen Elemente an den Anfang der Liste hinzu

Referenzen:
jQuery auseinandergenommen [http://www.keyframesandcode.com/resources/javascript/deconstructed/jquery/]
Die besten Einsätze von jQuery [http://net.tutsplus.com/articles/web-roundups/the-20-most-practical-and-creative-uses-of-jquery/]
Die 70 populärsten jQuery Plugins 2011 [http://stylishwebdesigner.com/70-most-popular-jquery-plugins-of-2011/]

Unobtrusive JavaScript

- Unobtrusive JavaScript bedeutet wörtlich **unaufdringliches** JavaScript
- Im Prinzip soll bei diesem Designmuster eine Seperation zwischen verschiedenen Ebenen (JS, HTML) erreicht werden
- Es geht darum, dass JavaScript den Funktionsumfang erweitern und keine Voraussetzung darstellen soll
- Dies geht über jQuery sehr leicht: Folgendes (altes) Verfahren (Auszug aus HTML):

```
01. <input type="date" onchange="validateDate();" />
```

- Dies wird nun durch unobtrusive JavaScript zu (Auszug aus HTML):

```
01. <input type="date" data-mindate="1980-01-01" data-maxdate="1990-01-01" data-validate="true" />
```

- Wie kann man das dahinter erreichen? Zum Beispiel durch folgenden JavaScript (in externer Datei!):

```
01. $(function() {
02.     $('input[data-validate="true"]').change(function() {
03.         /* Validierung unter Verwendung von data-Attributen */ });
04. });
```

Referenzen:
Unobtrusive JavaScript auf Wikipedia [http://en.wikipedia.org/wiki/Unobtrusive_JavaScript]
7 Regeln für Unobtrusive JavaScript [http://icant.co.uk/articles/seven-rules-of-unobtrusive-javascript/]
Seite über barrierefreihes JavaScript [http://ichwill.net/]

Beispiel (28)

HTML:

```
01. <input type="text" name="s" list="ex28-titles"
02.     data-autocomplete-source="http://html5skript.florian-rappl.de/search.aspx?callback=?" />
03. <datalist id="ex28-titles"></datalist><!-- Geht zur Zeit nur in Opera und Firefox! -->
```

JavaScript (sollte in externer Datei liegen!):

```
01. $(function() { //Verwenden von jQuery-Ready
02.     $('input[data-autocomplete-source]').each(function() { //Alle gefundenen Elemente durchgehen
03.         var target = $(this); //Kleiner Helfer
04.         target.keypress(function() { //Bei Veränderung machen wir was (Autocomplete über JSON)
05.             var options = {}; //Konstruieren das Optionen-Objekt
06.             options[target.attr('name')] = target.val(); //Zuweisen des Namens als Schlüssel, Wert als Wert
07.             $.getJSON(target.attr('data-autocomplete-source'), options, function(data) { //JSON Abfrage
08.                 content = []; //Buffer der Ausgabe
```

```
09.                 $.each(data.hits, function(i, v) { //Serverantwort durchgehen
10.                     content.push(v.title); //Und im Buffer speichern
11.                 }); //Ausgabe auf Seite (im datalist-Tag) speichern
12.                 $('#' + target.attr('list')).html('<option>' + content.join('</option><option>') + '</option>');
13.             });
14.         });
15.     });
16. });
```

Referenzen:
Tutorial für HTML5-Autocomplete [http://www.quackit.com/html_5/tags/html_datalist_tag.cfm]
W3Schools Informationen zum datalist-Tag [http://www.w3schools.com/html5/tag_datalist.asp]

ETags und Expired Header

- Entity-Tags sind ein Konzept das Server und Browser zur Validierung von Cache-Komponenten verwenden
- Im Cache vorhandene Elemente werden verwendet, wenn diese noch nicht abgelaufen sind
- Das Ablaufsdatum stellt man über den `Expires`-Header einer Serverantwort ein, z.B.

```
01. GET /images/beispiel.gif HTTP/1.1
02. Host: html5skript.florian-rappl.de
03. HTTP/1.1 200 OK
04. Last-Modified: Tue, 12 Dec 2012 03:30:59 GMT
05. Content-Length: 1195
```

- Das Verfallsdatum sollte möglichst weit in der Zukunft liegen (Resourcenabhängig!)
- Wenn die Resource veraltet ist, wird ein sog. *bedingter GET-Request* durchgeführt
- Serverseitige Behandlung ist entweder durch Vergleich des Last-Modified-Datums oder des ETags möglich
- ETags sind flexibler, da hier ebenfalls auf Änderungen in Headern eingegangen werden kann
- Der Nachteil von ETags liegt in der Verwendung von mehreren Servern (bzw. ein Cluster)
- Daher gilt: ETags sollten vermieden oder nur in korrigierter Fassung verwendet werden

Referenzen:
HTTP-Spezifikation bzgl. Expires [http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.21]
ETag bei Wikipedia [http://de.wikipedia.org/wiki/HTTP_ETag]
Gute Erklärung zu ETags und Caching von Marco Götze [http://solariz.de/2044/etag_webseite_cache_caching_beschleunigen.htm]

GZIP Komprimierung

- Die Performance kann durch einige HTML-Entscheidungen bereits deutlich verbessert werden
- Nichtsdestotrotz gibt es Faktoren wie die Bandbreite des Endanwenders, den ISP, die Nähe zu Peering-Punkten und andere auf die man keinen Zugriff hat
- Über eine Komprimierung des Antwortstreams (Response) kann man die benötigte Bandbreite verringern
- Dies hat mehrere positive Auswirkungen (schnellerer Zugriff, geringerer Traffic)
- Hierbei gibt es aber folgende 2 Nachteile zu beachten: Der Server muss mehr arbeiten (Komprimieren) und auch der Client (Dekomprimieren)

GZIP wird nur dann verwendet, wenn der Client ebenfalls mit dieser Technologie zurecht kommt

- Eine alternative Komprimierung bietet `deflate`, welche jedoch weniger effizient ist
- HTML-Dokumente, Skripten und Stylesheets sollten komprimiert werden, Image und PDF-Dateien nicht!
- Der Grund liegt ganz einfach darin, dass diese Formate bereits komprimiert sind und sich der Aufwand nicht lohnen würde eine Komprimierung durchzuführen

Referenzen:
Steve Souders über GZIP Komprimierung [http://developer.yahoo.com/performance/rules.html]
Artikel zur GZIP Komprimierung für die Homepage [http://blogschreiber.com/archiv/2009/12/quicktip-wie-man-seine-webseite-mit-gzip-optimiert-link/]
Die offizielle Seite von GNU zip (GZIP) [http://www.gzip.org/]
How To Optimize Your Site with GZIP [http://betterexplained.com/articles/how-to-optimize-your-site-with-gzip-compression/]

Inline Bilder und Sprites

- Zwei weitere Techniken um http-Anfragen zu sparen sind Inline-Bilder und CSS Sprites
- Erstere verwenden die bei Canvas besprochenen base64-kodierten Bytefolgen zur Speicherung des Bildes in HTML, z.B. wird aus der ersten Zeile (http-Request notwendig) die zweite (inline):

```
01. <img src="/icons/html5.png" />
02. <img src="..." />
```

- CSS Sprites bauen auf der bekannten Sprite-Technik auf - eine Folge von Bildern wird in ein (größeres) Bild gepackt
- Man nutzt dabei aus, dass man bei CSS Hintergrundbildern (`background-image`) über `background-position` die Position des Hintergrunds angeben kann, z.B.

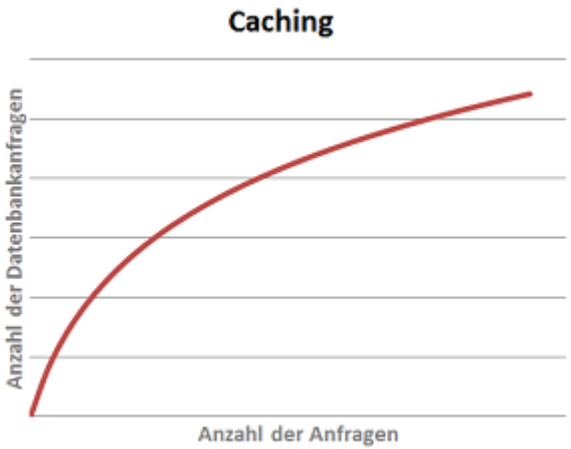
```
01. .flags {
02.    background: transparent url('flag-sprite.png') no-repeat; /* Gemeinsamer Hintergrund für ALLE */
03.    width: 16px; /* Gleiche Größe für alle */
04.    height: 16px; /* Elemente sehen in etwa so aus <div id="de" class="flags"></div> */
05. }
06. #cz { background-position: 0 -16px; } /* Verschiebt (0, 0) des Bildes auf (0, -16) des Elements */
07. #de { background-position: 0 -32px; } /* und mehr! */
```

- Viele Grafikprogramme beherrschen das Erstellen von Sprites (zus. gibt es Generatoren im Internet)

Referenzen:
Artikel zu Inline Images auf WebsiteOptimization [http://www.websiteoptimization.com/speed/tweak/inline-images/]
Base64 Encoded Images for Internet Explorer [http://dean.edwards.name/weblog/2005/06/base64-ie/]
Wikipedia Data URI scheme [http://en.wikipedia.org/wiki/Data_URI_scheme]
Binary File to Base64 Encoder [http://www.greywyvern.com/code/php/binary2base64]
CSS Sprits - What They Are, Why They're Cool, and How To Use Them [http://css-tricks.com/css-sprites/]
CSS Sprites um Ladegeschwindigkeit zu optimieren [http://webstandard.kulando.de/post/2010/05/26/css-sprite-ladegeschwindigkeit-optimieren-http-requests-verringern]
CSS Sprites: Image Slicing's Kiss of Death [http://www.alistapart.com/articles/sprites]

Serverseitiges Caching

- Hat man die Zahl der http-Requests auf ein Minimum gebracht und verwendet optimierte Skripten, CSS und weitere Resourcen (auch am HTML kann man sparen), so bleibt nur noch der Weg auf den Server
- Für statische Seiten ist man mit Komprimierung und sehr langen Expires-Header auf jeden Fall optimal aufgestellt
- Bei dynamischen Seiten liegt das Problem im Regelfall an der Zahl der getätigten Datenbankabfragen
- Daher sollte man auf jeden Fall sehr häufig benutzte Queries, die sich selten ändern, cachen



Der Effekt von guten Caching sollte auf jeden Fall spürbar sein

- Professionell gestaltete Webseiten verwenden einen Cache-Pool, der solche Resourcen in einen Zwischenspeicher (Arbeitsspeicher) hält
- Im Regelfall werden Resourcen mit einem Cache-Ablaufdatum und einer Priorität versehen, so dass veraltete Resourcen verworfen werden
- Auf der anderen Seite werden Resourcen je nach Prio auch verworfen, wenn der Speicher voll ist und Platz gemacht werden muss
- Caching ist bei allen großen Web-Skriptsprachen möglich (z.B. PHP, ASP, Ruby, ...)

Referenzen:
Codeproject Artikel [http://www.codeproject.com/KB/web-cache/MyCache.aspx]
Server Caching [http://www.caucho.com/resin-3.0/performance/caching.xtp]
Guide für Caching am Apache Server [http://httpd.apache.org/docs/2.1/caching.html]
Großes Caching Tutorial [http://www.mnot.net/cache_docs/]
Artikel für Caching mit PHP [http://rfc1437.de/page/caching-fur-php-systeme/]
MSDN Artikel für Caching mit ASP.NET [http://msdn.microsoft.com/de-de/library/bb979285.aspx]

JavaScript-Attribute

- Zwei besondere Attribute für den `<script>`-Tag wurden noch nicht betrachtet:
  - `async` (Skript asynchron laden und so bald als möglich ausführen) und
  - `defer` (nach hinten stellen und Reihenfolge beachten).
- Beide Elemente ermöglichen das Laden von JavaScript ohne Pausieren
- Die Möglichkeiten `<script src="..." async="true">` zu schreiben ist sehr neu – also noch keine ausgezeichnete Verbreitung
- Zusätzlich kann man noch noch ein `onload`-Attribut festlegen, welches ein Funktion nach dem erfolgreichen Ladevorgang ausführt
- Jeder `async`-Skript wird vor dem `window.load`-Ereignis (und sobald als Möglich) ausgeführt

- Bei `defer` wird die Reihenfolge auf jeden Fall eingehalten und der Skript noch vor dem `DOMContentLoaded`-Ereignis ausgeführt
- `defer` besitzt im Moment die deutlich bessere Unterstützung da es Teil der HTML 4 Spezifikation ist

Referenzen:
StackOverflow zu Async [http://stackoverflow.com/questions/1834077/which-browsers-support-script-async-async/6766189]
David Walsh über HTML5 async [http://davidwalsh.name/html5-async]
Microsoft-Test Drive für Async. Skripte [http://ie.microsoft.com/testdrive/Performance/AsyncScripts/Default.html]
MSDN über defer [http://msdn.microsoft.com/en-us/library/ms533719(v=vs.85).aspx]

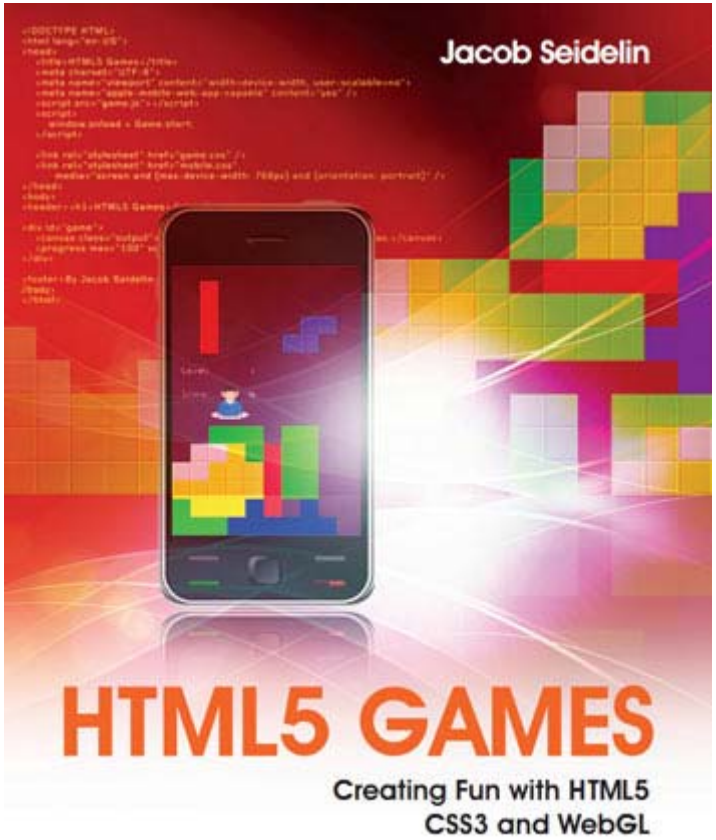
Beispiel (29)

```
<style>
.flags
{
    background: url('http://html5skript.florian-rappl.de/img/flagsprite.png') no-repeat;
    display: inline-block;
    height: 11px;
    padding-left: 20px;
}
#de { background-position: 0px 0px; }
#en { background-position: 0px -11px; }
#fr { background-position: 0px -22px; }
#jp { background-position: 0px -33px; }
#kr { background-position: 0px -44px; }
</style>
<script src="../../scripts/loadjquery.js" defer="defer"></script>
<img src="..." alt="HTML5 Logo" align="middle" /> HTML5 Beispiel!
<p>Bitte Sprache wählen: <a id="de" class="flags">deutsch</a> | <a id="en" class="flags">english</a> | <a id="fr" class="flags">français</a> | <a id="jp" class="flags">日本</a> | <a id="kr" class="flags">한국의</a></p>
```

Games mit HTML5

- Spiele mit HTML, CSS und JavaScript zu schreiben ist nichts neues
- Man kann z.B. `<div>`-Elemente animieren

- Einfache Objekte können durch SVG-Elemente dargestellt werden
- Mit HTML5 sind nun neue Technologien in den Bereichen Multimedia (Video, Audio, Canvas) und JavaScript (Selektor-API, WebSockets) hinzugekommen
- Außerdem besitzt CSS3 neue Möglichkeiten mit Transforms (und 3D-Transforms)
- Da einige dieser Technologien für Spiele noch nicht ausreichend sind, gibt es noch mehr
- So hat Google z.B. die Web Audio API[1] Spezifikation erarbeitet



Mittlerweile erscheinen auch allerlei Bücher zu dem Thema Spiele mit HTML5 Technologien erstellen

- Diese arbeitet effizienter als das <audio>-Element (erlaubt Mixing, mehrfaches / geplantes Abspielen, ...)

Referenzen:
Codeproject Artikel zu Spielen mit EaselJS [http://www.codeproject.com/KB/solution-center/HTML5-Gaming.aspx]
MSDN zu Spieleprogrammierung mit EaselJS [http://blogs.msdn.com/b/davrous/archive/2011/07/21/html5-gaming-animating-sprites-in-canvas-with-easeljs.aspx]
Ein XNA-Game (XboX etc.) auf Canvas portieren mit EaselJS [http://blogs.msdn.com/b/davrous/archive/2011/09/09/html5-platformer-the-complete-port-of-the-xna-

game-to-lt-canvas-gt-with-easeljs.aspx]
Die 30 besten HTML5 Spiele [http://www.casualgirlgamer.com/articles/entry/28/The-Best-30-HTML-5-games/]
Soundeffekte über HTML5 Audio [http://stackoverflow.com/questions/1933969/sound-effects-in-javascript-html5]

[1] Web Audio API [https://dvcs.w3.org/hg/audio/raw-file/tip/webaudio/specification.html]

Existierende Engines

- Eine Game Engine soll wiederkehrende Aufgaben für Spielen performant implementiert haben und den Zugang vereinfachen
- So sind Audio, Video, Eingabe und Regeln Bestandteile einer jeden Engine
- Man kann natürlich auch eine eigene Game Engine schreiben, jedoch ist die Verwendung eines existierenden Codes zeitsparend
- Außerdem erhält man somit eine vorhandene Seperation of Concerns - die Game Engine wird unabhängig weiterentwickelt und verbessert
- Verbesserungen an der Game Engine können ohne den vorhandenen Code ändern zu müssen übernommen werden
- Spiele mit HTML5 Technologien zu schreiben hat einen großen Vorteil: Zugänglichkeit
- Auf Smartphones, Tables, Fernsehern und Computern kann auf die Spiele über einen Browser (oder als native App verpackt) zugegriffen werden
- Neben freien Game Engines gibt es auch kommerzielle Projekte, die im Regelfall performanter sind und über einen regelmäßigeren Aktualisierungszyklus verfügen
- Der Hauptteil in der Spieleentwicklung fließt (dank einer externer Engine) in Grafiken und Soundeffekte

Referenzen:
Liste von Game Engines [https://gist.github.com/768272]
ImpactJS [http://impactjs.com/]
LimeJS [http://www.limejs.com/]
Aves Game Engine [http://ajaxian.com/archives/aves-game-engine]
Unity Game Engine [http://unity3d.com/]

Nützliche Bibliotheken

- Bei der Erstellung einer eigenen Game Engine empfiehlt sich die Verwendung von externen Bibliotheken
- Auch bei bestehenden Code sollte man zur Erweiterung bestehende Bibliothek in Betracht ziehen
- Um Sprites wunderbar zu animieren (auch Hintergründe etc.) empfiehlt sich Spritely[1], z.B.

```
01. $('#beispielelement').sprite({fps: 14, no_of_frames: 8});
```

  Dies benutzt ein CSS Spritesheet zur eigentlichen Animation.

- Häufig braucht man spezielle Benutzersteuerelemente (besonderes Design) - hierbei hilft jQuery v.a. in Verbindung mit jQuery UI ebenfalls[2,3], z.B.

```
01. $('.slider-handle').draggable({
```

```
02.     drag: function(e,ui) { /* Verändern des Wertes und optische Anpassungen */ }
03. });
```

- Mit Plugins wie Waypoints[4] kann man das Scrollverhalten des Benutzers auslesen, z.B.

```
01. $('.entry').waypoint(function() {
02.    alert('Du hast zu einem Eintrag gescrollt!');
03. });
```

Referenzen:

[1] Sprites mit jQuery [http://spritely.net/]
GameQuery [http://gamequery.onaluf.org/]
Anything Zoomer [http://css-tricks.com/anythingzoomer-jquery-plugin/]
Floater Plugin [http://www.designchemical.com/lab/jquery-floater-plugin/getting-started/]
FolderPreview [http://www.m2az.com/folderPreview/]

[2] Bunte Slider für jQuery [http://tutorialzine.com/2010/03/colorful-sliders-jquery-css3/]

[3] Bessere Checkboxen bauen [http://tutorialzine.com/2011/03/better-check-boxes-jquery-css/]

[4] jQuery Waypoints [http://imakewebthings.github.com/jquery-waypoints/]
Nützliche Plugins auf jQueryList [http://jquerylist.com/]
iGoogle Interface nachmachen [http://net.tutsplus.com/tutorials/javascript-ajax/inettuts/]

Beispiel (30)

Der verwendete Sprite - Zelda mit Laufrichtung

```
01. var zelda = $('#link'), world = zelda.parent(), direction = 0;
02. var logic = setInterval(function() { //Hier läuft die Logik ab (z.B. Bewegung)!
03.     if(direction == 1) zelda.css('bottom', '+=3');
04.     //usw. für alle Richtungen und PBC Check damit unser Held in der Welt bleibt!
05. }, 25);
06. var setAni = function(dir, press) {
```

```
07.     if(!press) { direction = 0; zelda.destroy(); } //Bewegt sich nicht mehr - Sprite stoppen
08.     else { direction = dir; zelda.sprite({ fps : 25, no_of_frames : 6 }).spState(dir); //Sprite starten
09. };
10. var keypress = function(event) {
11.     switch (event.keyCode) {
12.         case 37://Linke Pfeiltaste - 37 links, 38 hoch, 39 rechts 40 unten
13.             setAni(2, event.type !== 'keyup'); // Als Richtungen sind die einzelnen Zeilen des Sprites
14.             break;
15.     } //usw. für alle notwendigen Tasten
16.     return false;
17. }; /* verschiedene Events binden */
18. $('html').bind('keydown', keypress).bind('keyup', keypress); // Die beiden Events festlegen
```

Referenzen:
Sprite Database [http://sdb.drshnaps.com/]
Spriters Resource [http://spriters-resource.com/]
Ein Bild in CSS umdrehen [http://css-tricks.com/snippets/css/flip-an-image/]
Das Beispiel [http://html5skript.florian-rappl.de/offlineexample/zelda.html]

Die File API

- Mit dem `FileReader` und der File API sind Zugriffe auf Dateien möglich
- Es gibt ein Objekt `Blob`, der Inhalt aus dem Objekt `File` aus dem Objekt `FileList`
- Zunächst stellt sich die Frage, wie man Dateien auswählen kann
- Aus Sicherheitsgründen muss der Benutzer die Dateien auswählen – dies kann man folgendermaßen erreichen:
    - Klassisch über den `input` Typ `file`, z.B. auch mit `multiple`:

```
01. <input type="file" id="files" name="files[]" multiple />
```

    - Elegant und modern über Drag-and-Drop und das `dataTransfer.files` Objekt des Events, z.B.

```
01. function handleFileSelect(event) {
02.     var files = event.dataTransfer.files;
03.     /* Etwas damit machen */
04. }
```

- Anschließend kann man jede Menge Methoden verwenden um die Dateien auszulesen oder die Dateien per Ajax auf einen Server laden

Referenzen:
W3C Spezifikation zur File API [http://www.w3.org/TR/FileAPI/]
MDN Artikel: Dateien aus WebApplikationen verwenden [https://developer.mozilla.org/en/Using_files_from_web_applications]
File API mit Drag and Drop Demo [http://html5demos.com/file-api]
Wikipedia zur File API [http://en.wikipedia.org/wiki/HTML5_File_API]
HTML5 Labs über die File API [http://html5labs.interoperabilitybridges.com/prototypes/fileapi/fileapi/info]

Methoden im Detail

- Hat man die entsprechende(n) Datei(en) ausgewählt, kann man deren Inhalt auslesen
- Zunächst benötigt man dafür eine neue Instanz eines `FileReader`-Objektes
- Anschließend kann man folgenden Code verwenden:

```
01. var currentfile = evt.target.files[i];//Eine Datei aus der Auswahl auswählen
02. var reader = new FileReader();//Neue Instanz
03. reader.onload = (function(file) { //Anonyme Methode erstellt eigentliche Methode
04.     return function(e) {
05.         //Zugriff auf die aktuelle Datei des Readers über 'file'
06.         //Zugriff auf den Lesenvorgang über 'e.target.result'
07.     };
08. })(currentfile);//'currentfile' bezeichnet die Variable der aktuellen Datei
```

- Jetzt können wir das `reader`-Objekt verwenden um eine der folgenden Methoden auszuführen:
  - `readAsBinaryString(file)` liest die Datei als Binärobjekt ein mit `int`-Werten zwischen 0 und 255
  - `readAsText(file, encoding)` liest die Datei als Text ein - `encoding` ist hierbei optional
  - `readAsDataURL(file)` wandelt den Inhalt der Datei in eine DataURL um
  - `readAsArrayBuffer(file)` liest die Datei in eine `ArrayBuffer` Instanz ein

Referenzen:

Lokale Dateien mit JavaScript einlesen [http://www.html5rocks.com/en/tutorials/file/dndfiles/]
W3C Spezifikation zur File API [http://www.w3.org/TR/FileAPI/#filelist-section]
W3C Spezifikation des ArrayBuffer Elemens [http://www.khronos.org/registry/typedarray/specs/latest/]

Beispiel (31)

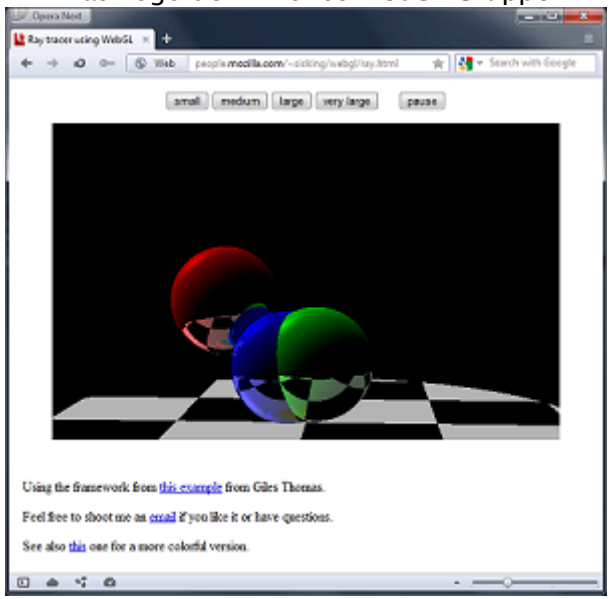
```
01. function handleFileSelect(evt) {
02.     /* Drag und Drop beenden */
03.     var files = evt.dataTransfer.files;// FileList Objekt verwenden
04.     var output = [];//Buffer für Ausgabe erstellen
05.     for (var i = 0, f; f = files[i]; i++) //Ein paar Eigenschaften auflisten (mit Formatierungen)
06.         output.push(f.name, f.type, f.size, f.lastModifiedDate.toLocaleDateString());
07.     document.querySelector('#ex31-list').innerHTML = output.join('');
08. }
09. //Die EventListener hinzufügen
10. var dropZone = document.querySelector('#ex31-drop_zone');
11. /* Eventlistener für dragover und dragleave festlegen */
12. dropZone.addEventListener('drop', handleFileSelect, false);
```

WebGL

- WebGL bietet 3D Grafik auf Opera (ab 12), Chrome (ab 9), Safari (ab 5.1), Firefox (ab 4) und IE (ab 11)
- Die Syntax ist dabei stark an OpenGL angelegt (basiert WebGL doch auf OpenGL ES 2.0)
- WebGL läuft im Canvas-Element des Browsers ab und kann Hardwarebeschleunigt werden
- Es gibt allerhand Demos[1] auf der Seite der Khronos WebGL Gruppe und Tutorials zu OpenGL[2]
- Folgender Code zum Initialisieren ist sinnvoll:

```
01. function initWebGL(canvas) {
02.   var gl = null;
03.   try { //WebGL Kontext holen
04.     gl = canvas.getContext("webgl") || canvas.getContext("experimental-webgl");
05.   } catch(e) {}
06.   if (!gl) alert("Der Browser unterstützt kein WebGL!");
07.   return gl; }
```

Das Logo der Khronos WebGL Gruppe



Abspielen des Ray-Tracing Demos auf Opera Next (v12)

Referenzen:

Cloudmach WebGL Platform für 3D Apps [http://cloudmach.com/]

Die WebGL Spezifikation [http://www.khronos.org/webgl/]
Offizielle WebGL Wiki [http://www.khronos.org/webgl/wiki/Main_Page]
[1] Demos zu WebGL [http://www.khronos.org/webgl/wiki/Demo_Repository]
[2] Tutorials zu OpenGL [http://www.opengl.org/sdk/docs/tutorials/]
Tutorial für WebGL [http://www.khronos.org/webgl/wiki/Tutorial]
MDN Tutorial zu WebGL [https://developer.mozilla.org/en/WebGL/Getting_started_with_WebGL]

Neue Attribute

- Über `contenteditable` kann man jedes Element als veränderbar kennzeichnen (mögliche Werte dafür sind `true`, `false`, `inherit`)
- Neu ist das `contextmenu` Attribut, welches auf die **ID** eines `<menu>`-Elements verweist
- Man braucht keine CSS `display: none;` Regel mehr wenn man `hidden="hidden"` verwendet
- Alle `data-*` Attribute sind frei wählbar und werden vom Browser ignoriert - perfekt für unobtrusive JavaScript
- Dank `draggable` kann man Elemente explizit als Dragbar markieren (`true`, `false`, `auto`)
- Um Drag and Drop einfacher zu machen gibt es auch noch `dropzone` mit den Werten `copy`, `move` und `link`
- Nicht zu verachten ist `spellcheck`, welches dem Browser mitteilt ob ein veränderbares Element durch die eingebaute Rechtschreibeprüfung nutzen soll (`true`, `false`)
- Das Attribut `xml:space` ist nun auch überall verfügbar und ermöglicht das Anzeigen von Leerzeichen über den Wert `preserve`
- Weiterhin vorhanden sind Attribute wie `accesskey` (Shortcut-Taste), `title`, `tabindex` (Reihenfolge), `dir` (`ltr`, `rtl` und `auto`) und `lang` (Sprache), sowie `id`, `class` und `style` für CSS

Referenzen:
HTML5-Attributes [http://www.htmlgoodies.com/html5/markup/html5-attribute-change-reference-for-web-developers.html#fbid=I1hlkhtIZeL]
xml:Space bei Sitepoint [http://reference.sitepoint.com/html/script/xmlspace]
Contenteditable Demonstration [http://html5demos.com/contenteditable]
W3Schools zu globalen Attribute in HTML [http://www.w3schools.com/html5/html5_ref_globalattributes.asp]

Interessante Attribute

- Wir haben bereits gesehen, dass das `<form>`-Element neue interessante Attribute erhalten hat, z.B.
  - `autocomplete` (soll die Autovervollständigung *[nicht]* verwenden?) und
  - `novalidate` (soll *[nicht]* nicht Clientseitig validiert werden?)
- Auch die Eingabeelemente haben einige neue Attribute erhalten - hier sind noch anzuführen:
  - `form` zur Auswahl des zuhörigen Formulars (entbindet Elemente vom übergeordneten Formular)
  - `list` für eigene Autocomplete-Listen (siehe Beispiel 28)
  - `multiple` (v.a. für `<select>` und `type=file` geeignet)

    - zusätzlich gibt es die bereits erwähnten Attribute
- Für das <menu>-Element (Kontextmenü) kann man mit dem Attribut `label` Sektionen erstellen
- Man kann Styles nun *scopen*, d.h. Gültigkeit nur für einen untergeordneten Block erklären, z.B.

```
01. <div>
02.     <style scoped="scoped"> /* Regeln */ </style>
03.     <!-- Noch mehr Elemente auf die diese Regeln zutreffen -->
04. </div>
```

Referenzen:
Artikel zu HTML5 data-* Attribut [http://css.dzone.com/news/html5-data-attribute-feed]
Eine Einführung in HTML5 Formulare [http://www.html5tuts.co.uk/tutorials/forms/]
28 HTML5 Features die man wissen muss [http://net.tutsplus.com/tutorials/html-css-techniques/25-html5-features-tips-and-techniques-you-must-know/]
HTML5-Attribute [http://www.htmlgoodies.com/html5/markup/html5-attribute-change-reference-for-web-developers.html#fbid=I1hlkhtIZeL]

Kleiner Texteditor

- Neben den erwähnten Attributen gibt es noch neue Attribute für <iframe>
- Diese dienen hauptsächlich der Sicherheitsverbesserung
- Mit dem sandbox Attribut kann man den Inhalt speziell behandeln, z.B. wird durch

```
01. <iframe src="search_form.htm" sandbox="allow-forms"></iframe>
```

  die Verwendung von Formularen in *search_form.htm* gestattet, der Rest (Skripte, Plugins, Zugriff auf Hauptseite etc.) ist untersagt
- Was man nicht außer Acht lassen sollte ist die vorher erwähnte Möglichkeit mit dem Attribut `contenteditable`
- Dadurch ist auch folgendes Möglich (einfach in die Eingabezeile (URL) kopieren):

```
01. data: text/html, <pre contenteditable>
```

- Diese URL (anscheinend eine DataURL) kann man auch Bookmarken und immer wieder aufrufen
- Diese Technik funktioniert auch mit JavaScript (z.B. `javascript: alert('hi!');`) und wird Bookmarklet (oder auch Favlet) genannt

Referenzen:
MSDN Artikel über die Erstellung eines Editors [http://msdn.microsoft.com/en-us/library/ms537834(v=vs.85).aspx]
Der HTML5 Editor Aloha [http://aloha-editor.org/]
StackOverflow Frage zu Editoren [http://stackoverflow.com/questions/4622403/any-wysiwyg-rich-text-editor-that-doesnt-use-html-contenteditable-or-designmod]
Das Web editierbar machen [http://universalruntime.tumblr.com/post/5244472335/contenteditable]

Slides von Paul Irish [http://dl.dropbox.com/u/39519/talks/html5-foundation/index.html#1]
Das Video des Talks von Paul Irish [http://www.youtube.com/watch?v=WxmcDoAxdoY]

Beispiel (32)

```
01. <ul contenteditable contextmenu="ex32-auswahl" id="ex32-list" onfocus="document.querySelector('#ex32-auswahl').hidden = '';"
    onblur="setTimeout(function() { document.querySelector('#ex32-auswahl').hidden = 'hidden'; }, 200);">
02. <li>Rotweinkuchen</li><!-- Eine einfache Liste - aber mit contenteditable (Wir erhalten neue Elemente!) -->
03. <li>Apfelkuchen</li> <!-- Die focus und blur Ereignisse wurden eingefügt um das Menü nutzen zu können -->
04. <li>New York Cheesecake</li> <!-- Später brauchen wir das nicht mehr - es wird als Kontextmenü angezeigt -->
05. </ul><!-- Menu funktioniert in noch KEINEM Browser -->
06. <menu id="ex32-auswahl" type="context" hidden="hidden"><!-- Hidden funktioniert (noch) nicht im IE -->
07. <button type="button" onclick="document.querySelector('#ex32-list').style.listStyleType = 'disc';">Rund</button>
08. <button type="button" onclick="document.querySelector('#ex32-list').style.listStyleType = 'square';">Eckig</button>
09. <button type="button" onclick="document.querySelector('#ex32-list').style.listStyleType = 'decimal';">Nummern</button>
10. <button type="button" onclick="document.querySelector('#ex32-list').style.listStyleType = 'none';">Ohne</button>
11. </menu>
```

Dirty, but valid HTML

- HTML5 zeichnet sich auch dadurch aus, dass die Validierung und das Fehlerverhalten standardisiert ist
- Dies erlaubt jedem HTML auf eine schnelle und kompakte Art und Weise zu schreiben
- Folgendes HTML-Dokument ist z.B. komplett vollständig (**und** validiert als HTML5!):

```
01. <!DOCTYPE html>
02. <meta charset="utf-8">
03. <title>Ein Beispiel</title>
04. <div>
05. Ein Test...
06. </div>
```

- Damit noch nicht genug - für Browser könnten wir sogar noch und Anführungszeichen weglassen
- Dies wird auch auf großen Seiten wie z.B. Google[2] praktiziert - hier werden keine Anführungszeichen verwendet
- Außerdem fehlt der <head> und <body>-Tag und die Paragraphs und der HTML-Oberknoten werden nicht geschlossen

- Neben solcher Erleichterungen muss man bei IDs und Klassennamen (Bezeichnern) jedoch immer noch auf Regeln aufpassen - daher im Zweifelsfall Tools wie Mothereffin[1] überprüfen

Referenzen:

[1] Mothereffin Seite [http://mothereff.in/]

[2] Google 404 [http://www.google.com/404]
Fancy ID Attribute [http://mathiasbynens.be/notes/html5-id-class]

Übersicht der Gefährdungen

- Die meisten Gefährderungen gehen von Inhalt aus, der von Benutzern eingebracht / verändert werden kann
- Die goldene Regel lautet: Trau **niemals** den Daten deiner Benutzer!
- Im Prinzip muss man sich beim Thema Sicherheit mit beiden Problemkindern beschäftigen:
  - Auf der Clientseite: Wie schreibe ich Markup, so dass dieser möglichst sicher ist?
  - Auf der Serverseite: Wie validiere ich Anfragen um die Echtheit der Daten sicherzustellen?
- Die HTML5 Features können bei Benutzereingeben oft negativ ausgenutzt werden, z.B. folgende

```
01. <form id="test" /><button form="test" formaction="javascript:alert(1)">X</button><!--formaction-->
02. <input onfocus=write(1) autofocus><!--autofocus-->
03. <video poster=javascript:alert(1)><!--poster, war zwar nur in Opera 10 vorhanden - dennoch beachten-->
04. <body onscroll=alert(1)><br><br><br><br><br><br>...<br><br><br><br><input autofocus><!--autofocus-->
```

- Auch in <source>-Elementen kann man über Ereignisse wie z.B. onerror JavaScript einbinden
- Daher gilt: Niemals Benutzer HTML-Elemente eingeben lassen - und wenn dies notwendig ist? Dann sollte man statt einer Blacklist (lange Liste mit vielen Regeln) eine Whitelist (kurze Liste) vorziehen
- Viele Sicherheitsprobleme entstehen durch JavaScript und damit verbundenes Cross-Site-Scripting (XSS)

Referenzen:
HTML5 Security Cheat Sheet [http://html5sec.org/]
Microsoft über XSS [http://support.microsoft.com/kb/252985]
Wikipedia zu Cross-Site-Scripting [http://en.wikipedia.org/wiki/Cross-site_scripting]
Artikel zu XSS auf Technical Info [http://www.technicalinfo.net/papers/CSS.html]

Offene Stellen

- Ob eine Sicherheitslücke ausgenutzt werden kann hängt oft vom Browsertyp und Browserversion ab
- Nichtsdestotrotz sollte man versuchen so viele Lücken wie möglich zu stopfen

- Da fast alle Lücken über Skripte kommen - und zwar nicht von den eigenen - sollte man bei Fremddaten immer vorsichtig sein
- Dies bedeutet, dass selbst Werbeeinblendungen, `<iframe>`-Tags und per JSONP geholte Daten in der Regel als kritisch betrachtet werden sollten
- Cookies sollten immer als `http-only` Cookies verschickt werden
- Daten sollten auf dem Client über `localStorage` bzw. falls diese temporär sind über `sessionStorage` abgespeichert werden
- Auch in `<object>`-oder `<embed>`-Tags können Skripte ausgeführt werden, z.B.

```
01. <object data="data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg==">
02. <embed src="data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTwvc2NyaXB0Pg==">
```

- Generell stellen Plugins wie Flash, Java etc. eine mögliche Gefahrenquelle dar (je nach Version und Verwendung) und sollten daher vermieden werden

Referenzen:
HTML5 Security Cheat Sheet [http://html5sec.org/]
Top 10 Attack Vectors [http://www.net-security.org/article.php?id=1656]
Die größten Lücken von HTML5 [http://www.infosecisland.com/blogview/18649-Top-Ten-HTML5-Attack-Vectors.html]
Die Gefahren von hybriden Apps und HTML5 [http://www.webvivant.com/feature-divide-and-conquer.html]

Immer UTF-8 verwenden

- Ein großes Sicherheitsmanko stellt die Nicht-Verwendung eines `<meta charset>`-Elements dar
- Dieses Element sollte das erste tragende Element sein und sofort nach `<html>` bzw. `<head>` stehen
- Kommt z.B. der `<title>`-Tag vor der Kodierung, und kann in diesem Tag Nutzerinhalt (z.B. der Benutzername) stehen, so ist Injection möglich!
- So könnte der Nutzer folgende Zeichenketten in den Tag-legen:

```
01. +ADw-script+AD4-alert(document.location)+ADw-/script+AD4-
```

- Die serverseitige Validierung würde in diesem Fall funktionieren, da dies wie ein normaler String aussieht
- Probleme ergeben sich wenn der Browser einfach zu UTF-7 greift, da noch kein Zeichensatz angegeben wurde
- Daher: **Immer** sowohl `<meta charset>` als auch die angegebene Kodierung (im Editor) wirklich verwenden!
- Niemals den Benutzer `<meta charset>`-Tags eingeben lassen oder dergleichen aus Ausgabe zulassen
- Alle Probleme ergeben sich erst durch die Verwendung von UTF-7 - daher gilt: Immer die Kodierung setzen und immer UTF-8 verwenden

Referenzen:
PHP Security über diese Lücke bei Wikipedia (ausgebessert) [http://blog.php-security.org/archives/38-Wikipedia-UTF-7-XSSCross-Site-Scripting-Hole-Plugged.html]
UTF-7 XSS Cheat-Sheet [http://openmya.hacker.jp/hasegawa/security/utf7cs.html]
Security Cheatsheet zu UTF-7 XSS [http://html5sec.org/#charset]

Beispiel (33)

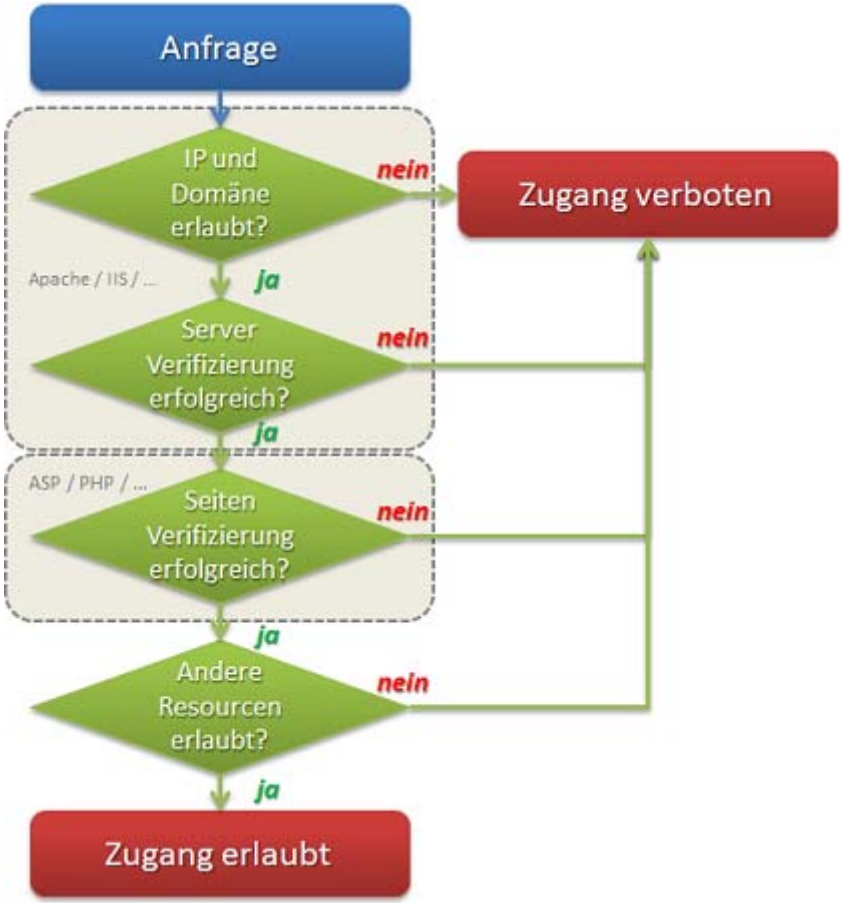
HTML vor dem Zugriff mit Benutzerdaten:

```
01. <img src="" id="ex33-img" />
02. <span id="ex33-txt"></span>
03. <iframe id="ex33-if" src="" frameborder=0></iframe>
```

HTML nach dem Zugriff mit Benutzerdaten

```
01. <img src="http://www.mysmilie.de/midi-smileys/boese/2/0040.gif" onload="alert('Du wurdest gerade gehackt!');" id="ex33-img" />
02. <span id="ex33-txt"><script>alert('Du wurdest gerade gehackt!');</script></span>
03. <iframe id="ex33-if" src="http://html5.florian-rappl.de/badiframe.html" frameborder=0></iframe>
```

Lecks am Server stopfen



Die Zugangsberechtigung zu Online-Resourcen muss über mehrerer Stufen hinweg gegeben werden

- Trotz aller Mühen werden die meisten Bedrohungen zwar von externen Quellen eingeschleust, jedoch von internen Quellen (Server) ausgegeben werden
- Daher ist es wichtig Zugänge zu sensiblen Resourcen nur über verschiedene Validierungen freizuschalten und Debugging-Informationen (Metadaten) niemals öffentlich anzuzeigen (nur lokal!)
- Bestimmte Informationen (Verzeichnisinhalte, bestimmte Dateitypen) dürfen nie angezeigt werden
- Nur eingeloggte Benutzer sollen Zugang zu bestimmten Möglichkeiten erhalten
- Nutzer sollen sich vor schreibenden Aktionen immer als solche Ausweisen müssen (Captchas etc.)
- Neben dem Schutz der eigentlich Seite und des eigentlichen Servers sollten auch andere Resourcen wie z.B. Datenbankserver entsprechend gesichert werden

Referenzen:

Wikipedia Artikel zur Authentifizierung [http://en.wikipedia.org/wiki/Authentication]
10 ways to make your network more secure and easier to manage [http://www.computerworlduk.com/how-to/infrastructure/36/10-ways-to-make-your-network-

more-secure-and-easier-to-manage/]
Strategies for Keeping a Secure Server [http://tldp.org/LDP/lame/LAME/linux-admin-made-easy/security.html]
Is dedicated server more secure than vps? [http://www.webhostingtalk.com/showthread.php?t=959446]
MSDN - Securing Your Web Server [http://msdn.microsoft.com/en-us/library/ff648653.aspx]
About - Secure Websites and Servers [http://webdesign.about.com/cs/websecurity1/a/aa061900a.htm]

Cross Site Request Forgery

- XSRF heißt soviel wie Seiten-übergreifende Aufruf-Manipulation und meint damit, dass dem Benutzer ein Seitenaufruf untergeschoben wird
- Geht man z.B. auf eine Seite auf der folgendes Quellcode platziert wurde,

```
01. <img src="http://de.wikipedia.org/w/index.php?title=Spezial:Userlogout" />
```

  so wird man sich anschließend von Wikipedia ausgeloggt wiederfinden - dies ist ärgerlich aber harmlos
- Wichtige Regel bei WebApplikationen: `GET` wird zum lesen von Daten verwendet und sollte immer das selbe Ergebnis liefern
- `POST` wird nur zum Schreiben von Daten verwendet und gibt unterschiedliche Ergebnisse aus - ein XSRF Angriff über eine falsche URL einer Resource kann durch notwendige `POST` Aufrufe verhindert werden
- In der Regel werden die Attacken so durchgeführt, dass man eingeloggt sein muss
- Ist man z.B. Administrator einer bestimmten (bekannten) Seite und geht (eingeloggt) auf eine andere Seite in der ein XSRF Angriff lauert, so kann einem ein Link zum Durchführen von bestimmten Adminaktionen wie z.B. Erstellen von Benutzern, Ändern von Benutzerpasswörtern etc. durchgeführt werden
- Zusätzlich sollte jedes Formular (d.h. jeder `POST` Request) durch ein Token (Verstecker Wert und Cookie) abgesichert werden - beim Posten werden die Werte des Cookies und des `hidden`-Eingabefeldes verglichen

Referenzen:
Wikipedia über XSRF [http://de.wikipedia.org/wiki/Cross-Site_Request_Forgery]
Open Web Application Security Project über XSRF [https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)]
So nutzen Hacker Cross Site Request Forgery für Angriffe [http://www.internet-magazin.de/ratgeber/erklaert-so-nutzen-hacker-cross-site-request-forgery-fuer-angriffe-213798.html]
Schutz vor Attacken durch Cross-Site-Request-Forgery ausgehebelt [http://www.heise.de/security/meldung/Schutz-vor-Attacken-durch-Cross-Site-Request-Forgery-ausgehebelt-6769.html]
Cross-Site Request Forgery [http://www.entwicklerblog.net/sicherheit/cross-site-request-forgery/]
ASP.NET MVC 3 Tutorial Video (Security) [http://www.pluralsight-training.net/microsoft/players/PSODPlayer?author=scott-allen&name=mvc3-building-security&mode=live&clip=0&course=aspdotnet-mvc3-intro]

Sichere Seiten mit SSL

- Um Seiten wirklich sicher zu machen braucht man SSL (d.h. ein SSL-Zertifikat)
- Dazu muss man sich zunächst ein solches Zertifikat kaufen, z.B. bei VeriSign, GeoTrust, GlobalSign oder einem anderen Anbieter
- Dank SSL wird die Verbindung zwischen Client und Server verschlüsselt - Grundlage ist ein asymmetrischer Schlüssel

- Die Performance der Webseite wird dadurch schlechter (auf Client- und Serverseite) - dafür ist die Verbindung sicher
- Ohne SSL sind Passwörter und andere sensible Daten als Klartext für jeden (Router) lesbar
- Ein anderer Nebeneffekt von SSL besteht darin, dass durch das Zertifikat die Seite als solche identifiziert wird
- Nur durch dieses Konzept kann man sichergehen, die Informationen gerade an die richtigen Leute zu geben
- Im Gegensatz zu SSL-losen Verbindungen wird nicht auf das `http` sondern auf das `https`-Protokoll gesetzt
- Unterschiedliche Browser lassen unterschiedliche Anbieter von Zertifikaten zur Überprüfung zu
- Soll das eigene Zertifikat auf allen Browsern verwendet werden, so ist der Kauf bei einem großen Anbieter empfehlenswert

Referenzen:
SSL Deutschland - Erklärung zu SSL [http://www.ssl.de/ssl.html]
Wikipedia Artikel zu SSL [http://de.wikipedia.org/wiki/Transport_Layer_Security]
Wikipedia Artikel über digitale Zertifikate [http://de.wikipedia.org/wiki/Digitales_Zertifikat]
Installation eines SSL Zertifikats auf dem Apache Server [http://www.digicert.com/ssl-certificate-installation-apache.htm]
Installation eines SSL Zertifikats auf dem IIS [http://support.microsoft.com/kb/299875]

W3C Specification

- Die W3C Spezifikation zu lesen kann viele Möglichkeiten aufzeigen und ist daher wichtig
- Als Beispiel betrachten wir die W3C Spezifikation der Geolocation[1]
- Die W3C Spezifikation wird niemals auf Unterschiede zwischen den Browsern oder den aktuellen Status der Implementierungen eingehen
- Am wichtigsten sind die hier definierten *Interfaces* z.B.

```
01. interface Coordinates {
02.     readonly attribute double latitude;
03.     readonly attribute double longitude;
04.     readonly attribute double? altitude; /* ... */
05. };
```

- Die Interfaces werden in einer IDL genannt CORBA beschrieben und geben uns einige Informationen über
  - Attribute und deren Status (`readonly` ist nur Get) sowie Typ (z.B. `double`)
  - Ob das Attribut immer vorhanden ist (`double?` entspricht einem Optional (`nullable`) Typ)
- Außerdem enthält die Spezifikation jede Menge (theoretischer) Beispiele und Erklärungen

Referenzen:

[1] W3C Spezifikation der Geolocation [http://www.w3.org/TR/geolocation-API/]
A List Apart [http://www.alistapart.com/articles/readspec/]

Artikel über W3C Design [http://www.germanforblack.com/articles/moving-towards-readable-w3c-specs]
W3Schools Einführung zu W3C [http://www.w3schools.com/w3c/w3c_intro.asp]
HTML5: Edition for Web Authors [http://www.w3.org/TR/html5-author/]
Wikipedia Artikel zu CORBA [http://en.wikipedia.org/wiki/OMG_IDL]
Wikipedia Artikel zu IDL [http://en.wikipedia.org/wiki/Interface_description_language]

Beispiel (34)

Manche US-Banken erlauben den Transfer über die URL, z.B. über folgenden Link

```
01. <a href="http://widelyusedbank.example.com?function=transfer&amount=1000&toaccountnumber=23234554333&from=checking">Hier
    klicken!</a>
```

Wer macht denn sowas? Leider ist die Antwort hierauf: Zu viele...

Wie können wir dieses lasche Verhalten nun ausnutzen? Wir gehen auf eine rießige Seite, die uns Kommentare zuläßt, welche Links beinhalten dürfen. Wir erstellen folgenden Post:

```
01. Wusstet ihr das man als Kunde von <i>widelyusedbank</i> <strong>immer</strong> eine Kontonummer erhält, welche die Quersumme 30
    hat?!
02. Schauts euch selbst an unter <a href="http://widelyusedbank.example.com">widelyusedbank website</a>.
```

Danach erstellt man mit einem zweiten (gefälschten) Account eine Antwort auf den ursprünglichen Beitrag. Diese könnte folgendermaßen aussehen:

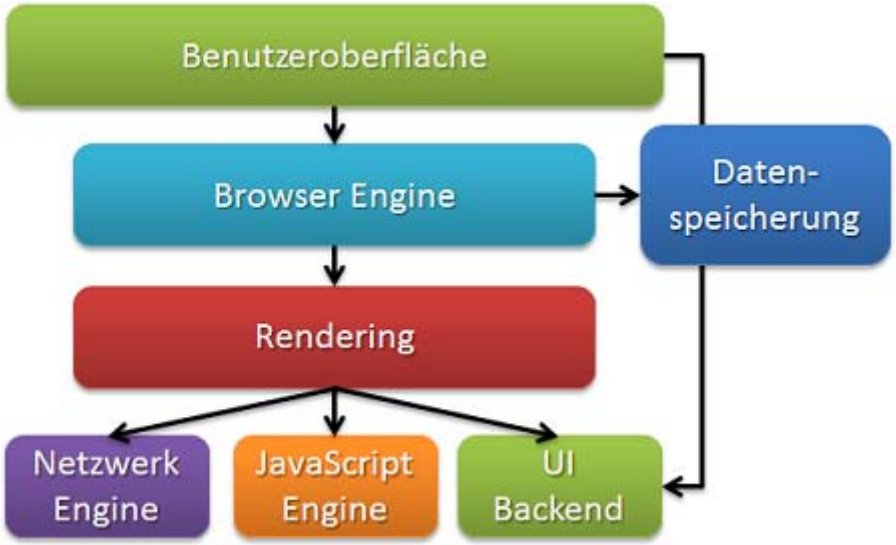
```
01. OMG das stimmt ja wirklich! Ziemlich seltsam...
02. <img src="http://widelyusedbank.example.com?function=transfer&amount=1000&toaccountnumber=23234554333&from=checking" />
```

Der Trick hier besteht darin, dass Kunden der Bank sich einloggen und danach nochmal den Kommentar lesen wollen. Dieser ist nun aber mit einem bösen Bild versehen, welcher den Transfer durchführt. (Zum Glück sind die mir bekannten deutschen Banken hier viel sicherer aufgestellt - allein schon wg. iTAN etc.)

Blick in den Browser

- Unser Code ist immer vollständig vom Browser abhängig
- Daher ist es wichtig sich über den Aufbau von Browsern im klaren zu sein
- Dabei ist das wichtigste Modul eines jeden Webbrowsers die eigentliche Browser Engine (z.B. Webkit, Gecko, Trident, ...)
- In der Engine werden alle Aktionen wie z.B. das Parsen von HTML oder CSS verwaltet

- Des Weiteren müssen sehr viele Resourcen gelesen werden können



Die 7 Hauptmodule eines modernen Webbrowsers zum Empfangen und Interpretieren von State of the Art Webseiten

- Für die eigentliche Anzeige ist die Rendering Engine verantwortlichen
- Ohne das Netzwerk-Modul könnten keine externen Resourcen und Webseiten empfangen werden
- Ganz unabhängig von den obigen Modulen ist die JavaScript Engine, welche die Interfaces des Browsers als in JavaScript ansprechbare Objekte freigibt
- Sie übernimmt auch die Ausführung und Interpretation (oder Kompilierung) der Skripte

Referenzen:
Life Of A Button Element [http://ontwik.com/javascript/txjs-2011-a9-alex-russell-life-of-a-button-element/]
Behind the Scenes of Modern Web Browsers [http://www.webappers.com/2011/08/19/behind-the-scenes-of-modern-web-browsers/]
HTML5 Rocks - Wie Browser funktionieren [http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/]
A Reference Architecture for Web Browsers [PDF] [http://www.grosskurth.ca/papers/browser-refarch.pdf]

Webkit Codebasis

```
01. namespace WebCore { // Namensraum für Webkit-HTML Objekte
02.     class CanvasPattern;
03.     class HTMLCanvasElement;
04.     /* Mehr class Prototypen */
05.     class CanvasRenderingContext { //Oberklasse für Context-Elemente
06.         WTF_MAKE_NONCOPYABLE(CanvasRenderingContext);
```

```
07.         WTF_MAKE_FAST_ALLOCATED;//Ausnutzen von Macros
08.     public: //Öffentlich zugängliche Methoden
09.         virtual ~CanvasRenderingContext() { }
10.         void ref() { m_canvas->ref(); }
11.         void deref() { m_canvas->deref(); }
12.         HTMLCanvasElement* canvas() const { return m_canvas; }
13.         virtual bool is2d() const { return false; }
14.         virtual bool is3d() const { return false; }
15.         virtual bool isAccelerated() const { return false; }
16.         virtual void paintRenderingResultsToCanvas() {}
17.         virtual bool paintsIntoCanvasBuffer() const { return true; }
18.     protected: //Nur für die Vererbung bestimmt
19.         CanvasRenderingContext(HTMLCanvasElement*); /* ... */
20.     private: //Nur für klasseninterne Verwendung bestimmt
21.         HTMLCanvasElement* m_canvas; //Referenz auf zugehöriges Canvas
22.         HashSet m_cleanURLs;
23.     };
24. }
```

- Webkit ist eine Browser engine basierend auf C++
- Apple Safari und Google Chrome verwendet Webkit
- Wer C++ kann wird sich hier zurecht finden
- Der Quelltext kann auch online angezeigt werden
- Alternativ kann man das Projekt runterladen und zuhause kompilieren
- Da es sich um ein großes Projekt handelt hilft es sich mit sog. Design Patterns auszukennen
- Das Projekt nutzt selbstverständliche Objektorientierte Programmierung (OOP) aus

Referenzen:
Offizielle Webkit Seite [http://www.webkit.org/]
Wikpedia Artikel zu Webkit [http://de.wikipedia.org/wiki/WebKit]
How Browsers Work - Architecture [http://www.vineetgupta.com/2010/11/how-browsers-work-part-1-architecture/]
Canvas 2D Rendering Context [http://codesearch.google.com/#OAMlx_jo-ck/src/third_party/WebKit/Source/WebCore/html/canvas/CanvasRenderingContext2D.h&type=cs]

Rendering nachvollziehen

- Wichtig ist die Phasen des Dokumentladevorgangs zu kennen:
    1. Dokumentanfrage
    2. Interpretation des Quelltextes inkl. DOM-Erstellung und laden weiterer Resourcen
    3. Erstellen des Layouts
    4. Die Seite zeichnen
- Ein zentrales Objekt ist das `RenderObject` (RenderObject.h) von dem Elemente erben:

```
01. class RenderObject : public CachedImageClient { /* RenderButton : public RenderDeprecatedFlexibleBox */
02.     friend class LayoutRepainter;               /* RenderDeprecatedFlexibleBox : public RenderBlock */
03.     /* Noch mehr friend class Protoypen */       /* RenderBlock : public RenderBox */
04. public:                                          /* RenderBox : public RenderBoxModelObject */
05.     RenderObject(Node*); /* Hängt an einem DOM-Node */    /* RenderBoxModelObject : public RenderObject */
06.     virtual ~RenderObject();
07.     RenderTheme* theme() const; /* Style-Regeln vom Betriebssystem! */
08.     /* ... */
09.     RenderObject* parent() const { return m_parent; } /* Im Tree! */
10.     /* ... */ };
```

- Es gibt jede Menge verschiedener spezialisierter `RenderObject` Klassen (rießiger Klassen-Baum)

Referenzen:
Video wie Gecko die Googleseite aufbaut [http://www.youtube.com/watch?v=nJtBUHyNBxs]
Rendering in WebKit [http://www.youtube.com/watch?v=RVnARGhhs9w]
WebCore Rendering I - The Basics [http://www.webkit.org/blog/114/]
RenderObject.h [https://github.com/WebKit/webkit/blob/master/Source/WebCore/rendering/RenderObject.h]
RenderButton.h [https://github.com/WebKit/webkit/blob/master/Source/WebCore/rendering/RenderButton.h]
Wikipedia Artikel zu Rendering [http://de.wikipedia.org/wiki/HTML-Rendering]

Beispiel (35)

Was passiert beim folgenden Code? (Wir ignorieren den Text - wäre ein einfaches `HTMLElement`)

```
01. <button>Das ist ein Button - mit Text</button>
```

Wir brauchen ein spezielles `HTMLElement` (Auszug aus *HTMLButtonElement.h*):

```
01. namespace WebCore {
02.     class HTMLButtonElement : public HTMLFormControlElement {
03.     public:
04.         HTMLButtonElement(const QualifiedName&, Document*, HTMLFormElement* = 0);
05.         virtual ~HTMLButtonElement(); // Destruktor
06.         virtual RenderObject* createRenderer(RenderArena*, RenderStyle*); /* ... */
07.         virtual bool canStartSelection() const { return false; } // Aha!
08.         // Hier stehen wirklich einzigartige Eigenschaften des Buttons drin!
09.         virtual bool willValidate() const { return false; } // Kommt uns bekannt vor
10.     private:
11.         enum Type { SUBMIT, RESET, BUTTON }; // Aha! /* weiteres Internes */
12.     };    }
```

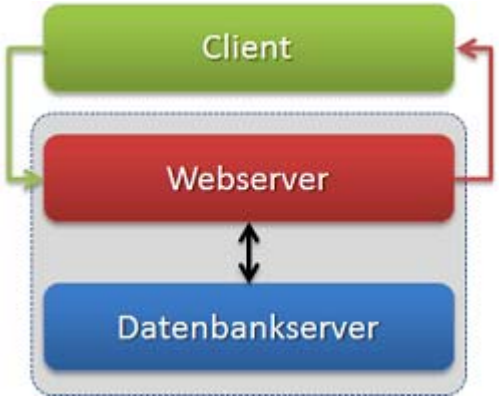
Betrachten wir mal die eigentliche Implementierung in *HTMLButtonElement.cpp* um das Rendering anzuschauen:

```
01. RenderObject* HTMLButtonElement::createRenderer(RenderArena* arena, RenderStyle*) {
02.     return new (arena) RenderButton(this); // Geben einen RenderButton als RenderObject zurück
03. }  //arena übernimmt das Speichermanagement (allocate und free Methoden) -- new ist überladen[1]
```

Referenzen:
HTMLButtonElement.cpp [https://github.com/WebKit/webkit/blob/master/Source/WebCore/html/HTMLButtonElement.cpp]
HTMLButtonElement.h [https://github.com/WebKit/webkit/blob/master/Source/WebCore/html/HTMLButtonElement.h]
HTMLButtonElement.idl [https://github.com/WebKit/webkit/blob/master/Source/WebCore/html/HTMLButtonElement.idl]

[1] new überladen in C++ [http://www.bearcave.com/software/c++_mem.html]

Interaktion perfektionieren

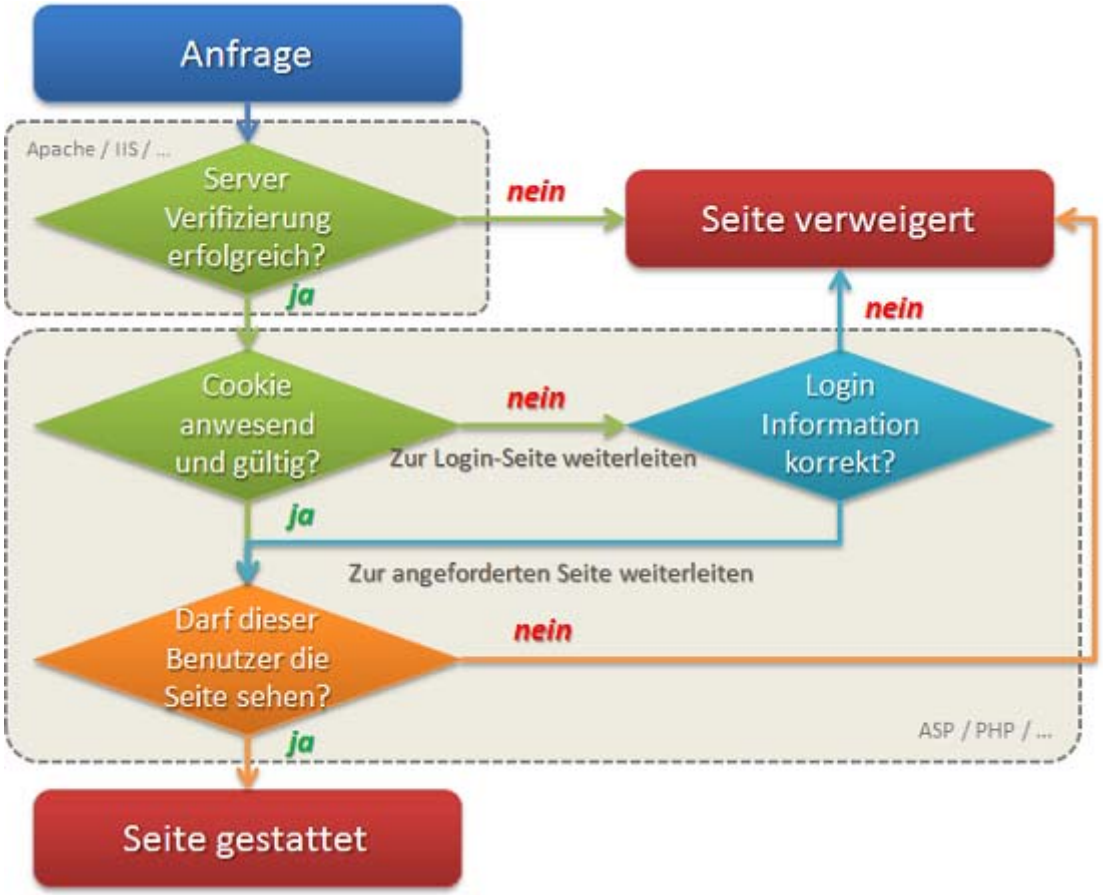


Beispielhafte Kommunikation

- Um eine anspruchsvolle WebApplication zu erstellen muss die Interaktion zwischen Server und Client reibungslos und ohne Fehler ablaufen
- Hierbei ist nicht nur die Kommunikation zwischen Client und Webserver, sondern auch die Kommunikation der einzelnen Dienste auf dem Server (Webserver zu Datenbankserver etc.) essentiell
- Je schneller die Kommunikation ablaufen kann desto mehr Anfragen können pro Sekunde beantwortet werden
- Am Client kann die Kommunikation durch AJAX, WebSockets, JSONP uvm. aufgebohrt werden
- Um die Seite entsprechend Nutzerfreundlich zu gestalten sollten Animationen nur sinnvoll z.B. zum Zeigen von neuen Inhalt oder zum Darstellen von Möglichkeiten eingesetzt werden
- Häufig gilt hier: Weniger ist mehr! Eine schlecht platzierte Animation macht mehr Schaden als keine
- Mögliche Interaktionen sollten entsprechend von statischen Inhalt abgetrennt werden - daher ist es nicht sinnvoll Links wie normalen Text zu formatieren
- Auch für Ajax-Get Requests sollten die URLs angepasst (und später ausgelesen) werden - und zwar über den Hash in `location.hash` (z.B. `#wert`) und das Ereignis onhashchanged

Referenzen:
Das Client/Server Modell [http://infomotions.com/musings/waves/clientservercomputing.html]
Allgemeine Client/Server Interaktion [http://penguin.dcs.bbk.ac.uk/academic/networks/application-layer/client-server/index.php]
Wikipedia Artikel zum Client/Server Modell [http://en.wikipedia.org/wiki/Client-server_model]
Einfache C++ Client/Server Beispiele [http://beej.us/guide/bgnet/output/html/multipage/clientserver.html]
The perfect web design app and why it doesn't exist [http://www.netmagazine.com/features/perfect-web-design-app-and-why-it-doesn-t-exist]
The Anatomy of a Perfect Web Site [http://johanbrook.com/minimalism/anatomy-perfect-web-site/]
Importance of Web Interactivity [http://www.hongkiat.com/blog/importance-of-web-interactivity-tips-and-examples/]

Validierungen einbauen



Ablauf der Benutzervalidierung mit Hilfe des serverseitigen Webseitencodes

- Neben den optionalen clientseitigen Validierungen sind serverseitige Validierungen immer notwendig
- Hierbei geht es aber um mehr als nur Formulareingaben
- Zum einen müssen Benutzer authentifiziert und autorisiert werden
- Zum anderen muss die Echtheit des Benutzers bestimmt werden
- Dies dient dazu um z.B. die erwähnten XSRF Attacken abzuwehren
- Des Weiteren sind Mechanismen wie die Verwendung von Captchas zu überprüfen
- Wichtig: Beim Weiterleiten die Zieladresse (URL) überprüfen

Referenzen:
Wikipedia Artikel zu Authentifizierung [http://de.wikipedia.org/wiki/Authentifizierung]
Wikipedia Artikel zu Autorisierung [http://de.wikipedia.org/wiki/Autorisierung]

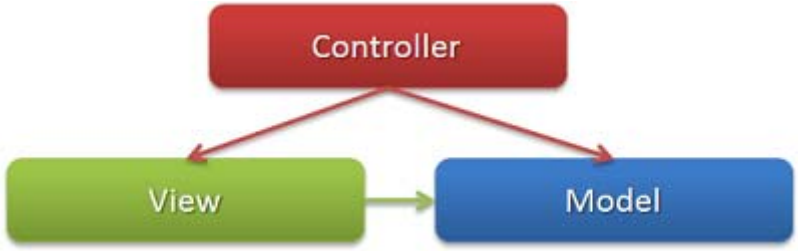
MSDN zu Authentifizierung und Autorisierung [http://msdn.microsoft.com/de-de/library/bb978972.aspx]
Datenvalidierung am Beispiel von ASP.NET MVC 2 [http://www.sascha-dittmann.de/post/ASPNET-MVC-2-Model-Validierung.aspx]

Designmuster für Webseiten

- Alles was wir in der Vorlesung gelernt haben hört sich gut an (und funktioniert auch gut)
- Die Frage ist jedoch – wie leicht läßt sich dies in der Realität an (großen) Webseiten umsetzen?
- Die Antwort hierauf ist: Es hängt davon ab! Und zwar wie man sich anstellt und welche Tools man verwendet
- Je nach Größe des Projektes und nach Ausrichtung muss man nicht zwingend so strukturiert vorgehen
- Ansonsten sind Software Design Pattern auch für Webseiten sehr zu empfehlen, v.a.
    - **MVC**, welches darauf abzielt eine starke Trennung zwischen Daten (Model), Webseite (View) und dem Code dazwischen (Controller) zu erzielen.
    - **MVP**, welches darauf abzielt eine normale Trennung zwischen Daten (Model), Webseite (View) und dem Code dahinter (Presenter) zu erzielen.
- Die beiden Konzepte klingen zwar ähnlich, jedoch ist MVP eine Weiterentwicklung von MVC und wird oftmals mit *Code-Behind* verstanden
- Beide Technologien erlauben den (leichten) Austausch der dahinterliegenden Ansichten
- In der eigentlichen Programmierung sollten dann Muster wie Factory Pattern, Builder Pattern, Command Pattern, Mediator Pattern oder Chain of Responsiblity verwendet werden

Referenzen:
Software Design Pattern [http://en.wikipedia.org/wiki/Software_design_pattern]
Design Pattern inkl. UML Diagrammen [http://www.dofactory.com/Patterns/Patterns.aspx]
Design Patterns in Web Programming [http://www.e-gineer.com/v1/articles/design-patterns-in-web-programming.htm]
Seite über prof. Softwareentwicklung [http://sourcemaking.com/]
An Introduction to Using Patterns in Web Design [http://37signals.com/papers/introtopatterns//index]

Model-View-Controller (MVC)



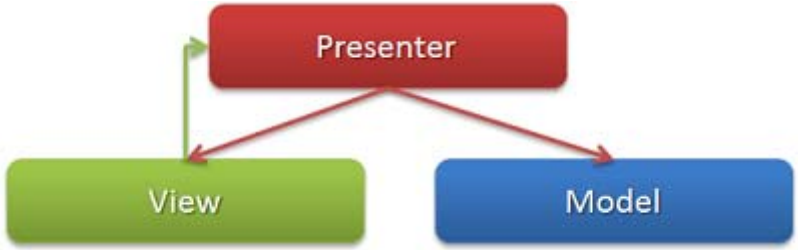
Beziehungen der Klassen mit direkten Verweisen im MVC Konzept

- Das MVC Konzept wurde bereits 1979 von Xerox bei der Entwicklung der ersten grafischen Benutzeroberfläche entwickelt
- Ziel war es ein gutes Konzept für die Trennung Daten / Oberfläche zu schaffen
- Das Model stellt die Daten der aktuellen Anfrage dar und soll sich ändern und validieren können

- Der View stellt das Model als Webseite dar und enthält alle möglichen Interaktionen mit der Datenquelle
- Der Controller nimmt die Benutzereingaben (URL, GET, POST) entgegen
- Er ist außerdem für die Füllung eines Models mit Daten und die Generierung eines entsprechenden Views verantwortlich
- Das Model ist nicht notwendigerweise eine Datenbank - im Regelfall wird es eine im Speicher abgelegte Instanz von Daten sein, die die Datengrundlage in einer Datenbank hat und Änderungen an dieser vollziehen kann
- MVC eignet sich bei Webanwendungen besonders gut, da man hier wirklich alle Steuerräder in der Hand hat

Referenzen:
Wikipedia Artikel [http://en.wikipedia.org/wiki/Model-view-controller]
MVC mit PHP [http://tutorials.lemme.at/mvc-mit-php/index.html]
Galileo Openbook zu MVC [http://openbook.galileocomputing.de/oo/oo_06_moduleundarchitektur_001.htm]
MSDN zu MVC [http://msdn.microsoft.com/en-us/library/ff649643.aspx]

Model-View-Presenter (MVP)



Beziehungen der Klassen mit direkten Verweisen im MVP Konzept

- Das MVP Muster wurde Anfang der 90er von Taligent (Apple, IBM, HP) entwickelt
- Es nimmt dabei MVC als Grundlage und kapselt die Funktionalität in einem Controller ähnlichen Objekt
- Zusätzlich dazu ist der View verantwortlich für die UI Ereignisse (Postback) - dies war eigentlich der Job des Controllers
- Das Model kapselt die Daten, welche dargestellt werden sollen
- Der View stellt die Daten in einer Webseite dar und empfängt die Ereignisse
- Die Ereignisse werden an den Presenter weitergeleitet, welcher dann entsprechende Aktionen durchführt
- Dabei werden Änderungen an den Model und den View durchgeführt
- Bei MVP ist in Verbindung mit WinForms und WebForms (.NET) sogar oftmals eine Portierung der Anwendung möglich
-

Referenzen:
Wikipedia Artikel [http://en.wikipedia.org/wiki/Model-view-presenter]
MVP mit ASP.NET [http://www.codeproject.com/KB/architecture/ModelViewPresenter.aspx]
ASP.NET MVP Webforms [http://webformsmvp.com/]

MSDN zu Model View Presenter [http://msdn.microsoft.com/en-us/magazine/cc188690.aspx]

Model-View-ViewModel (MVVM)

- Ein eher neues Konzept ist MVVM, das eine Verbesserung von MVP darstellt
- Das Konzept wird durch ereignisgesteuertes Binding ermöglicht - d.h. man bindet Darstellungen an Quellen
- Ändert sich eine Quelle (d.h. die Daten), wird die Darstellung aktualisiert
- Dies ist bei statischen Webseiten unmöglich, kann aber durch JavaScript ermöglicht werden

```
01. <p>First name: <input data-bind="value: firstName" /></p>
02. <p>Last name: <input data-bind="value: lastName" /></p>
03. <p>Full name: <strong data-bind="text: fullName"></strong></p>
```

- Eine bekannte Bibliothek hierfür ist KnockoutJS
- Hier werden verschiedene Elemente über unobtrusive JavaScript gebunden:

```
01. function AppViewModel() {
02.     this.firstName = ko.observable("Bert");
03.     this.lastName = ko.observable("Bertington");
04.     this.fullName = ko.computed(function() {
05.     return this.firstName() + " " + this.lastName();
06.     }, this);
07. }
08. ko.applyBindings(new AppViewModel()); // Activates knockout.js
```

Referenzen:
Dan Wahlin über MVVM [http://weblogs.asp.net/dwahlin/archive/2009/12/08/getting-started-with-the-mvvm-pattern-in-silverlight-applications.aspx]
KnockoutJS - JavaScript MVVM [http://knockoutjs.com]
Interaktives Tutorial zu Knockout [http://learn.knockoutjs.com/#/?tutorial=intro]
Stackoverflow Vergleich Knockout gegen Backbone [http://stackoverflow.com/questions/5112899/knockout-js-vs-backbone-js-vs]
Guido Mühlwitz über KnockoutJS [http://www.guido-muehlwitz.de/2011/08/knockout-js/]
Thomas Bandt über Lokalisierung mit KnockoutJS [http://blog.thomasbandt.de/39/2397/de/blog/clientseitige-lokalisierung-mit-net-und-knockoutjs.html]

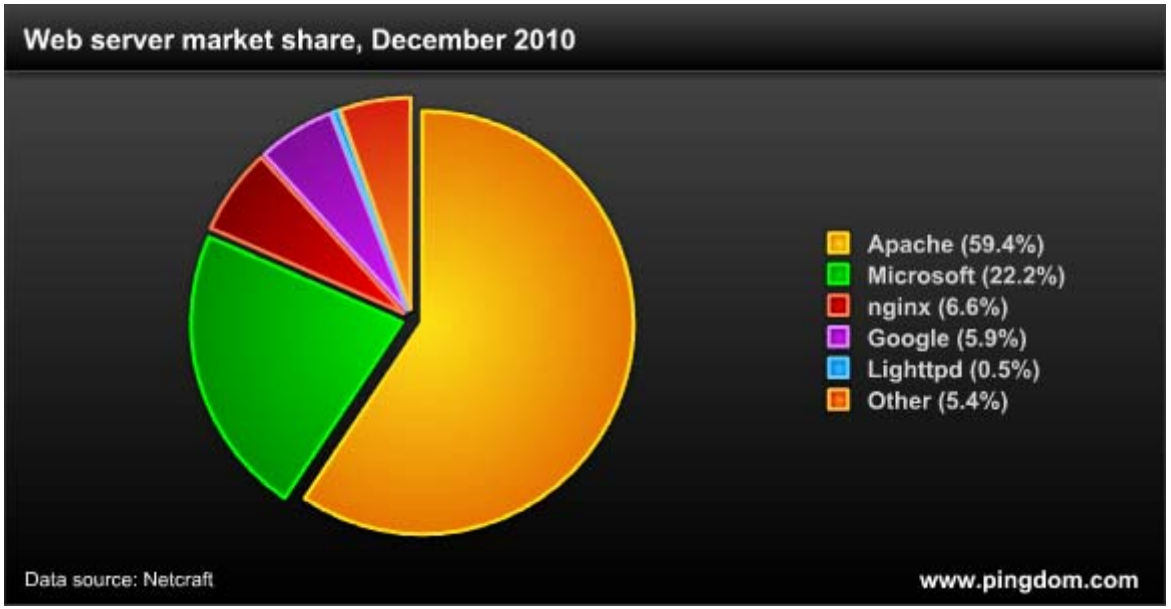
Bekannte Frameworks

- Das MVC Konzept eignet sich für Webseiten sehr gut, da Webseiten instanzlos sind und somit MVP nur mit Schwierigkeiten funktioniert
- Eine häufig gemachte Designentschiedung ist *Convention over Configuration* (CoC)

- Ein weiteres Konzept ist das der schnellen Entwicklung nach *Don't Repeat Yourself* (DRY)
- Für jede Websprache gibt es bereits vorhandene MVC Frameworks - einige Sprachen eignet sich besser für MVC als andere
- Eine Grundvoraussetzung für fortgeschrittenes MVC ist die OOP
- Bei **PHP** sind Frameworks wie Drupal (PAC), Joomla! Platform, Zend Framework oder Cake PHP recht beliebt
- Die Skriptsprache **Ruby** ist v.a. durch die MVC Frameworks Ruby on Rails und Nitro bekannt geworden
- Für **Python** gibt es z.B. das Framework Django oder Pyjamas (hier schreibt man serverseitig sowohl Python als auch JavaScript)
- Eine sehr große Palette an Möglichkeit hat man über ASP.NET MVC (3) mit **C#** - dies nutzt extensiv die Möglichkeiten von C# aus
- Das beinhaltet parallele Datenbankabfragen in der Programmiersprache (PLINQ), Extension Methods, Lambda Expressions, Dynamic, ...

Referenzen:
Wikipedia über Webbasierte MVC Frameworks [http://en.wikipedia.org/wiki/Model-view-controller#Implementations_of_MVC_as_web-based_frameworks]
Wikipedia Vergleich zwischen MVC Frameworks [http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks]
Wikipedia Artikel zu WebObjects (Java) [http://en.wikipedia.org/wiki/WebObjects]
Wikipedia Artikel zu Django (Python) [http://en.wikipedia.org/wiki/Django_(web_framework)]
Wikipedia Artikel zu ASP.NET MVC (C#) [http://en.wikipedia.org/wiki/ASP.NET_MVC_Framework]
Wikipedia Artikel zu Drupal (PHP) [http://en.wikipedia.org/wiki/Drupal]
Wikipedia Artikel zu Ruby on Rails (Ruby) [http://en.wikipedia.org/wiki/Ruby_on_Rails]

Server und Sprachen

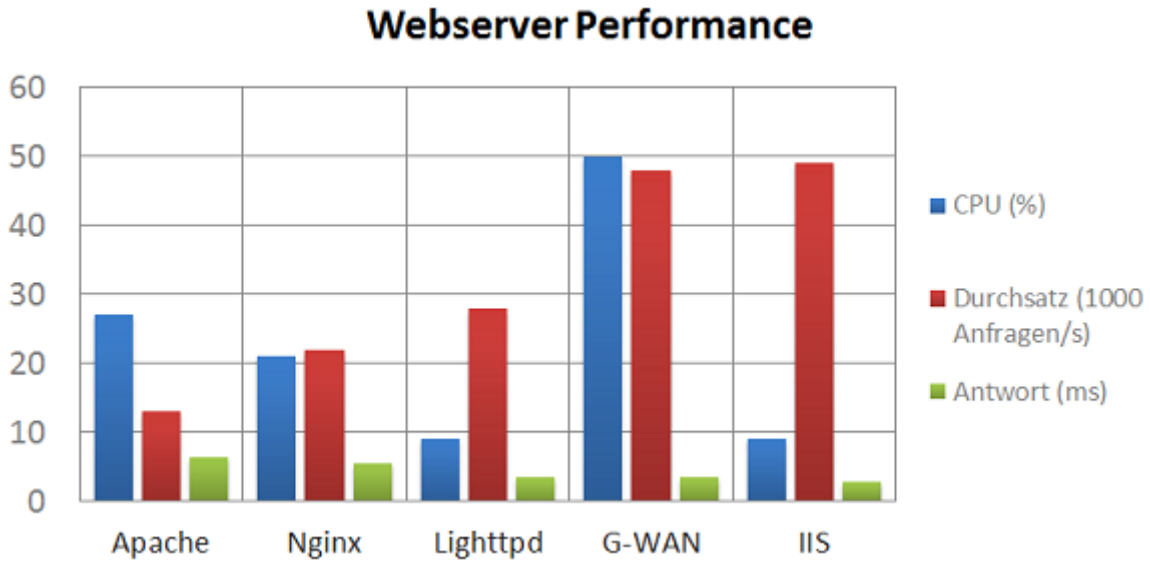


Die kostenfreien Linux Distributionen beherrschen den Webserver Markt

- Beim Thema Webserver hat man allerlei Auswahl
- Die Entscheidung fällt zwischen Windows und Linux (MacOS und andere haben geringere Verbreitung)
- Viele Firmen setzen auf Windows Server, da diese Aufgrund des gegebenen Supports Vorteile haben
- Bei günstigen Webhosting Angeboten erhält man im Regelfall einen Linux Server mit Apache
- Auch die Programmiersprache für Webseiten ist ein Gebiet mit viel Auswahl; hier stehen die Frage nach Performance und Größe / Ziel des Projektes im Vordergrund
- Für Skriptsprachen spricht die hohe Verbreitung an kompatiblen Servern und die schnelle Entwicklungszeit
- Für Hochsprachen spricht die schnelle Performance und die exzellente Erweiterbarkeit

Referenzen:
Wikipedia Artikel über Mac OS X Server [http://en.wikipedia.org/wiki/Mac_OS_X_Server]
PHP gegen Perl [http://www.thesitewizard.com/archive/phpvscgi.shtml]
Active Web Sites and Comparison of Scripting Languages [http://training.gbdirect.co.uk/courses/php/comparison_php_versus_perl_vs_asp_jsp_vs_vbscript_web_scripting.html]
Scripting Language Comparison Chart [http://www.mybestratedwebhosting.com/best-web-hosting-tips/scripting-language-comparison-chart.html]

Vergleich der Performance



Der IIS (7) schneidet in allen Bereichen sehr gut ab während Apache nur im Mittelfeld liegt- G-WAN ist sehr gut allerdings mit hoher CPU Belastung

- Einen unabhängigen, fairen Vergleich von Server und Skriptsprachen zu erhalten ist nicht möglich
- Jeder Server und jede Skriptsprache ist für andere Zwecke gebaut worden (*The best tool for the job*)

- Die freien Skriptsprachen wie PHP, Ruby, Perl, Python eignen sich sehr gut für kleine bis mittlere Projekte
- Bei größeren Projekten muss man schon zu allerhand Tricks greifen und sich sehr sehr gut auskennen
- Ansonsten empfiehlt sich die Verwendung von Sprachen wie Java, C# oder gar C/C++

Referenzen:
Despite enterprise dominance, Microsoft struggles in Web server market [http://arstechnica.com/business/news/2011/09/despite-enterprise-dominance-microsoft-struggles-in-web-server-market.ars]
The Fastest Webserver? [http://www.webperformance.com/load-testing/blog/2011/11/what-is-the-fastest-webserver/]
Why Many Developers Hate ASP.NET and Why They're Wrong [http://net.tutsplus.com/articles/editorials/why-many-developers-hate-asp-net-and-why-they-are-wrong/]
Seite von G-WAN [http://gwan.com/]
G-WAN gegen Nginx [http://www.wikivs.com/wiki/G-WAN_vs_Nginx]
Which server to use for small static files [http://nbonvin.wordpress.com/2011/03/24/serving-small-static-files-which-server-to-use/]

Beispiel (36)

- Wir folgen dem Beispiel von Nerddinner[1] – die Konzepte sind umgesetzt durch `Routes` (legen URLs fest),

```
routes.MapRoute("Default", "{controller}/{action}/{id}", //Namen der Route und Aufbau dieser festlegen
                new { controller = "Home", action = "Index", id = "" }); //Standard-Werte festlegen
```

- `Controller` (steuern die Ansichten und Modelle),

```
[Authorize] //Attribute steuern besondere Eigenschaften der Methoden z.B. nur eingeloggte Benutzer
public ActionResult Edit(int id) { //Die Methode heißt wie eine URL - Parameter wie Query
    Dinner dinner = dinnerRepository.GetDinner(id);//Holt einen Eintag aus der Datenbank
    if (!dinner.IsHostedBy(User.Identity.Name)) // Fragt ob ob dieser dem Benutzer gehört
        return View("InvalidOwner"); // Falls nicht wird auf einen View der InvalidOwner heißt geleitet
    return View(dinner); //Ansonsten wird der Standardview angezeigt mit den Daten aus der DB
}
```

- `Models` (stellen Daten dar) und `Views` (Ansichten – hier am Beispiel der Razor Syntax)

```
<h2>The planets in our Solar System</h2><ul>
@foreach(var p in Model) {
    <li>@Html.ActionLink(p.Name, "Index", new { p.Id })</li>
}</ul>
```

Referenzen:

[1] Nerddinner bei Codeplex [http://nerddinner.codeplex.com/]
Wikipedia zur Razor Syntax [http://en.wikipedia.org/wiki/Microsoft_ASP.NET_Razor_View_Engine]
Scott Gu über die Razor Syntax [http://weblogs.asp.net/scottgu/archive/2010/07/02/introducing-razor.aspx]

Node.js

Das offizielle Logo von node.js

- Node.js oder kurz node ist Google's V8 Engine Standalone (ohne Browser)
- Zusätzlich zur Engine gibts noch einen entsprechenden Kontext (analog `Window`)
- In diesem Kontext sind Prozess spezifische Variablen und Umgebungseigenschaften
- Dadurch kann man JavaScript nun auch als allgemeine (GP) Skriptsprache ansehen
- Es ist sehr leicht eigene Programme / Code mit JavaScript interagieren zu lassen
- Dies hat bereits für eine unzählige Anzahl an node.js Modules geführt
- Wichtig für uns sind Module die auf das Dateisystem und Netzwerk zugreifen

Referenzen:
Die offizielle node.js Seite [http://nodejs.org/]
Download node.js [http://nodejs.org/]
Node.js als Shell [http://www.2ality.com/2011/12/nodejs-shell-scripting.html]
Tutorial für node.js [http://www.nodebeginner.org/]
Wikipedia Eintrag zu node.js [http://en.wikipedia.org/wiki/Nodejs]

Die Node API

- Nach dem Starten von node sitzt man im allgemeinen Kontext
- Hier hat man zunächst (scheinbar) weniger Möglichkeiten als im vorher bekannten `Window` Kontext

- Dies täuscht – tatsächlich offenbart dieser Kontext sehr viele Systemmöglichkeiten
- Nach wie vor existiert `console` (nur wird jetzt wirklich das Terminal bedient)
- Über `require` wurde das Modulsystem integriert (bekannt aus RequireJS)
- Neue Objekte wie `Int8Array` wurden hinzugefügt
- Die Eigenschaft `process` ist unser Zugang zum eigenen Prozess
- Hier ist z.B. die Prozess-ID `pid` enthalten (`process.pid`)
- Die Startargumente sind über `process.argv` zugreifbar (Array, 1. Element ist node, 2. der Skript, ...)
- Die geladenen Module sind in `module` verfügbar (z.B. ist *repl* geladen)

Referenzen:
Node.js API [http://nodejs.org/api/index.html]
Process Objekt in node.js [http://nodejs.org/api/process.html]

JavaScript am Server

- Mit dem *http* Modul kann man einen Webserver starten
- Als einfachstes Beispiel kann man *Hallo Welt* bei Anfragen als Text zurückgeben:

```
01. var http = require('http');
02. http.createServer(
03.   function (request, response) {
04.     response.writeHead(200, {'Content-Type': 'text/plain'});
05.     response.end('Hallo Welt\n');
06.   }
07. ).listen(8000);
```

- Natürlich kann man noch mehr machen (Parameter auslesen, dynamische Rückgabe, ...)
- Analog kann man natürlich auch Anfragen senden, z.B. folgender *GET* Request

```
01. var http = require('http');
02. http.get("http://www.google.com/index.html", function(res) {
03.   console.log("Got response: " + res.statusCode);
04. }).on('error', function(e) {
05.   console.log("Got error: " + e.message);
06. });
```

Referenzen:
Hilfe zum HTTP Modul [http://nodejs.org/api/http.html]
Stackoverflow: Webserver mit node.js [http://stackoverflow.com/questions/6084360/node-js-as-a-simple-web-server]

View engines

- Bevor man tiefer in serverseitige Webseiten mit JavaScript einsteigt sollte man sich für eine view engine entscheiden
- Eine view engine abstrahiert die Rückgabe (Text, i.d.R. HTML), so dass man keine seltsamen Strings mehr bauen muss
- Die meisten view engines basieren darauf, dass man HTML schreibt, welches Platzhalter enthält, die gefüllt werden
- Manche Systeme abstrahieren sogar noch diese Stufe, so dass man sogar kein HTML mehr schreiben muss
- Als Beispiel kann man *Jade* anführen:

```
01. doctype 5
02. html(lang="en")
03.   head
04.     title= pageTitle
05.   body
06.     h1 Jade - node template engine
07.     #container.col
08.       if youAreUsingJade
09.         p You are amazing
10.       else
11.         p Get on it!
```

Referenzen:
Verschiedene view engines für node.js [http://stackoverflow.com/questions/1787716/is-there-a-template-engine-for-node-js]
Die Jade view engine [http://jade-lang.com/]

Stackoverflow: View engines für ASP.NET MVC [http://stackoverflow.com/questions/1451319/asp-net-mvc-view-engine-comparison]
Channel9 über view engines [http://channel9.msdn.com/coding4fun/articles/Developer-Review-Four-ASPNET-MVC-View-Engines]

Anwendungen mit JavaScript

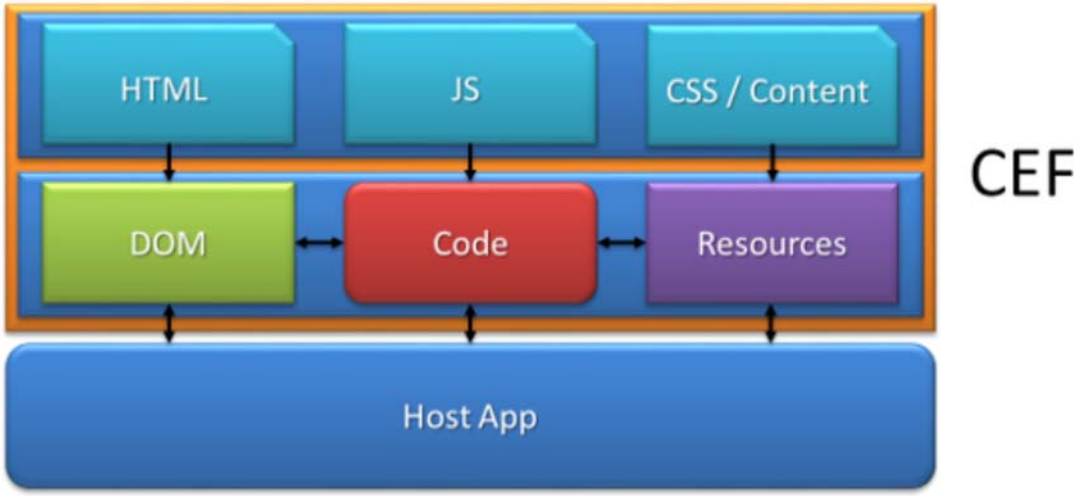
- Wenn JavaScript mit node am Server läuft - ist dies auch am Client möglich?
- Die Antwort ist: selbstverständlich!
- Möglichkeit 1: Mitgabe der node.js Runtime (V8, ...) bei Installation
- Möglichkeit 2: Bündelung der JavaScript Quellen als ausführbare Datei
- Nachteil von 1 ist, dass der Quellcode einsehbar und die Installation größer als Notwendig ist
- Bei Möglichkeit 2 kann man Updates nicht mehr schön Nachreichen und wird sehr spezifisch
- Das Beste wäre daher ein Hybride - eine native Anwendung, die den Ladeprozess startet und JS Dateien dynamisch lädt
- Diese könnten dann auch entsprechend geupdated werden, wären aber für Benutzer direkt uneinsehbar

Eine einfache Anwendung die komplett mit JavaScript geschrieben wurde

Referenzen:

V8 Projektseite [http://code.google.com/p/v8/]
TideSDK [http://www.tidesdk.org/]
Stackoverflow: Diverse Frameworks [http://stackoverflow.com/questions/109399/can-you-do-desktop-development-using-javascript]
JavaScript für Cliententwicklung [http://www.codeproject.com/Articles/501735/The-Spectre-Framework]
Das EnyoJS Projekt [http://enyojs.com/]

Das Chromium Embedded Framework



Architektur einer Anwendung mit dem Chromium Embedded Framework

- Abschließende Frage: Was bringt einem JavaScript wenn man Platform gebunden ist?
- Antwort: Nichts, weshalb man die Benutzeroberfläche auch abstrahieren muss
- Die naheliegendste Abstraktion ist HTML, d.h. wir brauchen einen Renderer
- Geschickte Lösung: Das Chromium Embedded Framework - quasi Chrome ohne UI
- Der große Vorteil: Man kann Webseiten direkt als Anwendungen extrahieren und Server/Client zusammenbringen

Referenzen:
Projektseite des Chromium Embedded Framework [http://code.google.com/p/chromiumembedded/]
Adobe zu The Chromium Embedded Framework [http://blogs.adobe.com/webplatform/2013/05/01/the-chromium-embedded-framework/]
Wikipedia Artikel [http://en.wikipedia.org/wiki/Chromium_Embedded_Framework]
Chromium Projektseite [http://www.chromium.org/Home]
What is the CEF? [http://kevinboyle.ie/blog/2013/05/16/what-is-chromium-embedded-framework]
Seite des Spectre Projekts (.NET Port) [https://github.com/crystalbyte/spectre]
CefSharp (bekannterer .NET Port) [https://github.com/cefsharp/CefSharp]

Literatur

Die folgenden Bücher habe ich zur Zusammenstellung des Stoffes verwendet.

- Uwe Hess, Günther Karl: **HTML 4**, bhv Verlag 2000, ISBN 3-8287-5005-2
- Bruce Lawson, Remy Sharp: **Introducing HTML5**, New Riders 2011, ISBN 0-3216-8729-9
- Steve Souders: **High Performance Websites**, O'Reilly 2008, ISBN 3-8972-1850-5
- jQuery Community Experts: **jQuery Cookbook**, O'Reilly 2010, ISBN 0-5961-5977-1

- Florence Maurice: **CSS3 Leitfaden**, Addison-Wesley 2010, ISBN 3-8273-3031-4
- Ed Tittel, Norbert Mikula, Ramesh Chandak: **XML For Dummies**, mitp 2000, ISBN 3-8266-2860-8
- David Siegel: **Web Site Design**, Markt und Technik 2000, ISBN 3-8272-5331-4
- Emily Vander Veer: **JavaScript For Dummies**, Wiley Publishing 2005, ISBN 0-7645-7659-3
- Richard Mansfield: **CSS Web Design For Dummies**, Wiley Publishing 2005, ISBN 0-7645-8425-1
- Christoph Prevezanos: **Jetzt lerne ich HTML5: Start ohne Vorwissen**, M+T 2011, ISBN 3-8272-4674-5
- Jesse Skinner: **Unobtrusive Ajax**, O'Reilly 2007, ISBN 0-5965-1024-4