

HTML5

Dr. Günter Kolousek

2012-09-17

Übersicht

- 1** Allgemeines
- 2** Grundlagen
- 3** Semantisches HTML
- 4** Multimedia
- 5** HTML5 Formulare
- 6** Offline-Webanwendungen
- 7** Geolocation

Geschichte von HTML

- 1 HTML, 1989 von Tim Berners-Lee (CERN)
 - SGML (aber: kein Browser, eigener Parser)
- 2 HTML 4.01, 1999 (W3C: <http://w3.org>)
- 3 XHTML 1.0, 2000 (W3C)
 - XML
- 4 WHATWG, 2004
 - Web Hypertext Application Technology Working Group (Apple, Mozilla, Opera): <http://whatwg.org>
 - Web Applications 1.0 → HTML5
- 5 XHTML 1.1, 2010 (W3C)
- 6 HTML5, XHTML5 (W3C Arbeitsgruppe, geplant: 2014)
- 7 HTML, “living standard” (WHATWG)

Probleme

- Browserkrieg
 - eigene Implementierungen
 - Browser diktieren HTML-Praxis
- XHTML 2 (2009 von W3C eingestellt)
 - klare Struktur, Trennung: Präsentation von Inhalt
 - WHATWG: “nicht praktikabel”
- W3C vs. WHATWG
 - WHATWG: “W3C: träge, bürokratisch”

Ziele von HTML5

- Ausrichtung auf Webapps
- evolutionäre Weiterentwicklung (Kompatibilität)
- Vereinheitlichung, Vereinfachung
 - hinzufügen, abändern, entfernen von Elementen, Attributen
- Barrierefreiheit
 - Teile von WAI-ARIA integriert
 - Web Accessibility Initiative - Accessible Rich Internet Applications Suite
 - `role` und `aria-*` Attribute

Webpage, Website, WebApp, MobileApp

■ Website

- besteht aus Webpages

- Dokumente!

- statisch oder dynamisch

■ Web application framework (↷ Server!)

- dynamische Website

- Webapplication (Webapps)

■ Webapp

- Anwendung, die über das Internet zugreifbar ist
- unterschiedliches Layout und UI zu Website!
- Dynamik: Server vs. Client vs. Server+Client

■ Native Application

- mobiles Gerät vs. PC

■ MobileApp

- Webapp oder native App für das Smartphone, Handy,...
- neue Features (Geolocation, Beschleunigungssensor,...)

Kriterien zur Auswahl

- zeitliche Änderung der Information
- Anforderungen an die Latenz bei Benutzereingaben
- benötigte Features
 - z.B. Zugriff auf lokale Geräte und installierte Programme
- Deployment
 - Installation: Online vs. Offline (CD,...)
 - Vertrieb: Direkt (offline, online) vs. Store (AppStore, Google Play,...)
- Wartung und Updates
- Portabilität
- Online vs. Offline
- Sicherheit
- Unabhängigkeit (siehe Stores...)

Begriff und Abgrenzung 1

- im engeren Sinne: Weiterentwicklung von HTML 4.01
- HTML5
 - definiert DOM (Document Object Model)
 - legt genaue Regeln fest (wie Verarbeitung stattfinden soll)
 - legt APIs fest
- HTML5 aus der Sicht des W3C
 - Semantische Elemente
 - XHTML und XML Einbettung
 - HTML/DOM Erweiterungen
 - Drag und Drop API
 - Video und Audio
 - Application Cache
 - HTML Formularerweiterungen
 - WAI-ARIA

Begriff und Abgrenzung 2

- HTML aus der Sicht der WHATWG
 - HTML5 des W3C
 - +Microdata
 - zusätzliche Informationen in HTML
 - Attribute: `itemscope`, `itemprop`, `itemid`, `itemref`
 - +Canvas2D Context
 - +WebSockets
 - +WebStorage
 - +WebWorker
- HTML5 aus erweiterter Sicht
 - HTML der WHATWG
 - +File API
 - +Geolocation API
 - +SVG
 - +MathML

Die guten Browser

■ Desktop

- Firefox 3.0+
- Safari 3.0+
- Opera 10.0+
- Chrome 3.0+
- Internet Explorer 9+ (alte IE!!!)
 - am Besten auf IE10 zu warten!!!!

→ <http://caniuse.com>, <http://html5test.com>

■ Mobile → <http://mobilehtml5.org/>

Unterricht

- Chrome 30+
- HTML5 aus erweiterter Sicht **mit** Unterstützung in Browsern
- keine Rücksichtnahme auf “alte” Browser!
 - In Produktion: \leadsto “feature detection”
 - \leadsto <http://www.modernizr.com/>
- Details in der Spezifikation bei WHATWG
 - <http://www.whatwg.org/specs/web-apps/current-work/multipage/>
- “Übersichtliche” Darstellung des Source-Codes
 - vs. Komprimierung zwecks Performance-Optimierung
- muss gültig sein! \leadsto HTML5-Validator
<http://validator.w3.org>
- Zusätzliche Links: <http://w3schools.com>,
<http://diveintohtml5.info>, <http://html5rocks.com/>

“Unverändertes” - 1

q Kurzes Zitat Wie schon Bertold Brecht sagte: `<q>Erst kommt das Fressen, dann die Moral.</q>`

samp Ausgabe eines Programmes `<p>The System answers: <samp>Too deeply nested!</samp></p>`

kbd Benutzereingabe Bitte geben Sie ein:
`<kbd>scp
www.example.com:/etc/password
anon@www.myserver.com</kbd>`

sub, sup `<var>x₀</var>²`

div, span Ohne Bedeutung

br Zeilenumbruch

ul, ol, li Listen

“Unverändertes” - 2: Tabellen

```
1 <table>
2   <caption>Schuelernoten</caption>
3   <thead>  <!-- optional -->
4     <tr><th>Schueler</th><th>Noten</th></tr>
5   </thead>
6   <tbody>  <!-- optional -->
7     <tr><td>Mini</td><td>1</td></tr>
8     <tr><td>Maxi</td><td>5</td></tr>
9   </tbody>
10  <tfoot>
11    <tr><td>Schnitt</td><td>3</td></tr>
12  </tfoot>
13 </table>
```

Hello World 1

Möglich, aber nicht bei uns (Bug im IE \leadsto body)!

```
1 <!DOCTYPE html>
2 <HEAD>
3     <META charset="utf-8">
4     <title>Hello</title>
5 </HEAD>
6 <h1>Hello HTML5
7 <p>Hello
8 <p>World!
```

Hello World 2

Gleicher DOM wie vorher

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="utf-8">
5    <title>Hello</title>
6  </head>
7  <body>
8    <h1>Hello HTML5</h1>
9    <p>Hello</p>
10   <p>World!</p>
11 </body>
12 </html>
```

Aufbau 1

- Element
 - Start-Tag, Attribute, Inhalt, Ende-Tag
 - leere Elemente, z.B. `<meta charset="utf-8"/>`
 - bestimmte Anfangs- und Endetags optional
- Attributwerte in doppelten oder einfachen Anführungszeichen
 - unter bestimmten Bedingungen auch keine Anführungszeichen
- Attributwert optional, z.B.
`<input type="checkbox" checked>`

Aufbau 2

- 1 Byte Order Mark (BOM, U+FEFF, optional)
- 2 Leerzeichen und Kommentare (optional)
- 3 Doctype
- 4 Leerzeichen und Kommentare (optional)
- 5 `html` Element (mit Inhalt, optionales Start- bzw. Endetag)
- 6 Leerzeichen und Kommentare (optional)

XHTML5

- XML Syntax!
- Auslieferung mittels HTTP Header “Content-Type”
 - `text/xml`, `application/xml` oder `application/xhtml+xml` (anstatt `text/html`)!!
- XML Deklaration (Doctype optional)
- `xmlns` Attribut für `html`
- `<noscript>` von JavaScript-Zustand unbeeinflusst!
- `document.write()` bzw. `.writeln()`: keine Funktion
- Zeichensatz via XML Deklaration (`meta` wird ignoriert!)
- nicht in IE 6 bis 8 (↪ Download-Dialog)!

Unsere Regeln für die Notation

- HTML5 (kein XHTML5)
- Zu jedem öffnenden Tag ein schließendes
 - außer leere Elemente
- alle Element- und Attributnamen in Kleinbuchstaben
- Jeder Attributwert in Anführungszeichen (vorzugsweise doppelte)
- `class` und `id`:
 - beginnen mit Buchstaben
 - dann: Buchstaben, Ziffern, `_`, `-`, `.` und `:`
- Zeichensatz **immer** UTF-8 und `meta` Element
 - innerhalb der ersten 512 Bytes
- Komprimierung
 - <http://code.google.com/p/htmlcompressor/>
 - auch JavaScript und CSS (auf Basis von YUI Compressor)
 - gute Komprimierung für Javascript: <http://slimit.org/>

Verschiedene Änderungen - 1

■ Elemente

- **Entfernt:** `frame`, `frameset`, `noframes`, `tt`, `big`, `acronym`, `applet`, `center`, `dir`, `font`, `isindex`, `strike`
- **nicht empfohlen:** `document.write()` und `.writeln()`
- **neu, z.B.:** `audio`, `video`, ..., `wbr` (Zeilenumbruch möglich)

■ Attribute

- **neue globale:** `class`, `accesskey`, `tabindex`, `id`, `lang`, `hidden`, `data-*`

■ Standard für `type`

- `script` hat automatisch `type="text/javascript"`
- `style` hat automatisch `type="text/css"`

Verschiedene Änderungen - 2

- `<a>`
 - ohne `href` ist korrekt! \leadsto lediglich Platzhalter!
 - `download me`
 - dzt. **nicht** im W3C Standard, aber Unterstützung in Browsern!
 - damit keine Server-Unterstützung mehr notwendig!
 - darf alles außer interaktive Elemente beinhalten:

```
1 <a href="a.html">
2   <h1>A header</h1>
3   <p>A paragraph</p>
4 </a>
```

- `img:alt` nicht immer zwingend!
- `blockquote` muss kein `<p>` beinhalten
- XPath-Variante (99% XPath 1.0 und etwas XPath 2.0)

Verschiedene Änderungen - 3

- Attribut `rel` gibt Art der Verlinkung an
 - bei `<a>`, `<link>`, `<area>`
 - HTML definiert Werte von `rel` neu
 - Wichtige Werte (tw. Browserunterstützung)
 - `alternate` ... alternative Repräsentation des Dokumentes
 - `author` ... Author des Dokumentes
 - `help` ... Hilfe
 - `icon` ... Icon für aktuelle Seite (nicht für `a`, `area`)
 - `license` ... Lizenz
 - `nofollow` ... Autor meint: nicht folgen (nicht bei `link`)
 - `noreferrer` ... keinen HTTP Header `Referer` schicken (nicht bei `link`)
 - `prefetch` ... soll vorgeladen werden
 - `search` ... kann zum Durchsuchen der Seite verwendet werden
 - `stylesheet` ... importiere Stylesheet (nicht für `a`)

data-* - Datenspeicher

- HTML5-konforme Möglichkeit neue Attribute hinzuzufügen
- \leadsto Daten je Element speichern
- Attribut in jedem Browser verwendbar
 - \leadsto graceful degradation: Webseite bleibt bedienbar, auch wenn Browser das Feature nicht unterstützt!
 - da Browser unbekannte Attribute ignorieren!
- JavaScript - mäßiger Zugriff
 - über `attributes...`
 - über `dataset` Property (außer IE)

```
1 <div id='donauturm' data-height='252m'></div>
2
3 <script>
4     var building = document.getElementById('donauturm');
5     var height = parseInt(building.dataset.height);
6
7     building.dataset.height = height + 1 + "m";
8 </script>
```

Content Model

Kein Blockelement und Inlineelement mehr, sondern

- Metadata: Metadaten, z.B. `title`
- Flow: Inhalt, z.B. `p`, `img`, `strong` oder Text
- Sectioning: Semantische Elemente für Dokumentgliederung, z.B. `section`, `nav`
- Heading: Überschriften, z.B. `h1`, `hgroup`
- Phrasing: Auszeichnungen, z.B. `em`, `span`, `img`
- Embedded: Externe Inhalte, z.B. `iframe`, `video`
- Interactive: Nutzerinteraktion, z.B. `a`, `input`

Aufbau 3

```
1  <!DOCTYPE html>
2  <html> <head>
3      <meta charset="utf-8">
4      <meta name="author" content="Guenter Kolousek">
5      <meta name="description"
6          content="Kurzbeschreibung fuer Google...">
7      <meta name="keywords"
8          content="von, google, wahrscheinlich, ignoriert">
9      <title>Hello</hello>
10     <!-- fuer relative links -->
11     <base href="http://mustermann.com/maxi/index.html">
12     <link rel="stylesheet" type="text/css" href="css/main.css">
13 </head> <body>
14 <h1>Hello HTML5</h1> <p>Hello, World!</p>
15 <script src="script.js"/>
16 </body> </html>
```

■ \leadsto DOM

■ <http://developer.yahoo.com/performance/rules.html>

Semantisches HTML

- Dem Inhalt einer Webseite eine Bedeutung geben!
- Struktur in eine Webseite bringen
 - `section, header, footer, aside, nav, article:`
Outline-Algorithmus
 - anstatt verschachtelter `div`
 - \leadsto `hgroup`
- Textpassagen
 - `strong, em, dfn, code, var, address, cite, dfn, mark`
- Maschinenlesbarer gestalten (Suchmaschinen,...)
 - Microdata-Format
 - Ruby-Annotationen (Aussprache z.B. japanischer Schriftzeichen)

Inhaltliche Gliederung

```
1  <section>           <!-- Sinnabschnitt, wie z.B. Kapitel -->
2    <h1>Kapitel 1</h1>
3    bla bla
4  </section>
5  <section>
6    <h2>Kapitel 2</h2>
7    <p>bla bla</p>
8    <section>
9      <h1>Kapitel 2.1</h1>
10     bla bla
11   </section>
12 </section>
```

Weiters: `article` für geschlossene Inhalte (z.B. Blog, News)

Kopf- und Fußbereiche von Abschnitten und Dokumenten

```
1 <section>
2   <h1>Kapitel 1</h1>
3   bla bla
4   <footer>bla bla</footer>
5 </section>
6 <section>
7   <header>
8     <h2>Kapitel 2</h2>
9     Willkommen im Kapitel 2
10  </header>
11  <p>bla bla</p>
12  <section>
13    <h1>Kapitel 2.1</h1>
14    bla bla
15  </section>
16 </section>
```

Hauptnavigation und ergänzende Inhalte

```
1  <section>
2    <header>
3      <h1>Maxi Mustermann</h1>
4      <a href="impressum.html">Impressum</a>
5      <nav>
6        <ul>
7          <li><a href="home.html">Startseite</li>
8          <li><a href="about.html">Ueber uns</li>
9        </ul>
10     </nav>
11   </header>
12   <p>bla bla</p>
13   <aside>
14     <q>People ask me what I do for fun when I'm not at work.
15     But I'm paid to do my hobby, so I never know what to
16     answer.</q>
17   </aside>
18 </section>
```

Überschriften und Abschnitte

```
1 <h1>Level 1</h1>
2 <section>
3   <h1>Level 2</h2>
4   <section>
5     <h1>Level 3</h2>
6     <section>
7       <hgroup> <!-- nur h1 bis h6 erlaubt -->
8         <h3>Uebertext, z.B.: "Sensation vom Tag"</h3>
9         <h1>Level 4, eigentlicher Titel</h1>
10        <h4>Untertitel</h4>
11      </hgroup> <!-- nur h1 im Outline-Algorithmus -->
12    </section>
13  </section>
14</section>
```

Semantische Textauszeichnungen 1

strong wichtiger Inhalt (nicht mehr starke Betonung)

em betonter Inhalt (stärker durch Verschachtelung)

b vom restlichen Text abgesetzt, typographische Darstellung normalerweise fett, z.B. Produktname, Anreisstext.

Kein “bold”; weder wichtig noch betont; zum restlichen Text gleichwertig;

i wie **b**, aber normalerweise kursiv, z.B. Fachbegriffe, Definitionen.

s nicht mehr relevanter Inhalt; “durchgestrichen”, z.B. “Statt-Preise”.

del entfernte Inhalte; “durchgestrichen”.

small “Kleingedrucktes”

hr inhaltlicher Bruch; z.B. Themenwechsel; “horizontale Linie”

Semantische Textauszeichnungen 2

dfn Definition, `<dfn title="Begrenzung eines HTML-El."> HTML-Tag</dfn>`

abbr Abkürzungen und Akronyme (kein acronym!)
`<abbr title="Höhere Techn. Irranstalt"> HTI</abbr>`

code gleich, aber Sprache angeben (empfohlen):
`<code class="language-python">1+1</code>`

var Variable im allgemeinen Sinn (Platzhalter), z.B.
`<var>Alice</var> sendet <var>n</var> Nachrichten an <var>Bob</var>`

address Adressinformation für das Dokument bzw. einen Artikel (nicht für allgemeine Adressen).

dl Beschreibungslisten, z.B. FAQs (vs. Definitionen)

cite Titel eines Werkes, z.B. Wikipedia Artikel
`<cite>Esperanto</cite>` beschreibt die am weitesten verbreiteten Plansprache.

Semantische Textauszeichnungen 3

time Zeit und Datum

```
1 <time>22:33</time>  
2 <time datetime="2012-08-09">morgen</time>
```

- 2012, 2012-W46, 2012-08, 2012-08-31, 08-31
- 13:15, 13:15:59, 2012-08-31T13:15, 2012-08-31T13:15.555
- Z (wenn UTC), +01:30, -0245, 2012-08-31T13:15Z, 2012-08-31T13:15-0200

mark einfach nur hervorgehoben, z.B. Suchergebnis

progress Fortschrittsbalken

```
1 <progress max="5" value="2">  
2   Schritt 2 von 5  
3 </progress>
```

Semantische Textauszeichnungen 4

meter Anzeige von Werten zwischen Min- und Max-Wert

```
1 <meter min="0" max="100"  
2     low="25" high="75"  
3     optimum="50" value="33">33</meter>
```

figure inhaltlich alleinstehende Teile auf die verwiesen wird, z.B. Tabellen, Graphen, Codebeispiele

```
1 <figure id="sales">  
2     <figcaption>Umsaetze 2012</figcaption>  
3       
4 </figure>
```

img - Element:

- alt="": reines Designelement
- alt fehlt, dann von figcaption!
- alt="text": soll Inhalt beschreiben
 - wenn keine figcaption oder dieses das Bild nicht ausreichend beschreibt!

Bildformate

- JPEG
 - beste Komprimierung (verlustbehaftet)
 - für Photos
- PNG-8 (portable network graphics)
 - “Nachfolger” der GIF Datei
 - verlustlose Komprimierung
 - max. 256 Farben
- PNG-24
 - max. 16 Millionen Farben
 - Alphakanal
- Optimizer: <http://optipng.sourceforge.net/>
- Sprites: <http://de.spritegen.website-performance.org/>

Audio 1

```
1 <audio id="player" src="audio.mp3" controls>
2   <a href="audio.mp3">audio.mp3 herunterladen</a>
3 </audio>
```

oder

```
1 <audio id="player" src="audio.mp3" controls>
2   <source src="audio.mp3">
3   <source src="audio.ogg">
4   <a href="audio.mp3">audio.mp3 oder... herunterladen</a>
5 </audio>
```

Audio 2

Wichtige Attribute:

controls Steuerbuttons; wenn fehlt und JavaScript aktiviert, dann ist Element unsichtbar.

autoplay beginnt automatisch

loop beginnt wieder von vorne

muted default-mäßig stumm

```
1 document.getElementById("player").onclick = function() {  
2     new Audio("audio.mp3").play();  
3 }
```

Audio implementiert das DOM Interface `HTMLAudioElement`, das von `HTMLMediaElement` (obige Attribute).

Video 1

Wie bei Audio, z.B.

```
1 <video controls autoplay muted poster="vorschau.jpg">
2   <source src="video.mp4">
3   <a href="video.mp3">video.mp4 herunterladen</a>
4 </video>
```

- Bleibt allerdings auch bei fehlenden `controls` sichtbar!
- Video implementiert `HTMLVideoElement`, das auch von `HTMLMediaElement` abgeleitet ist \leadsto Attribute
- Weiteres Attribut `preload` (von `HTMLMediaElement`)
 - `none` Preloading ist nicht anzuraten
 - `metadata` Preloading ist akzeptabel
 - `auto` Browser kann preloaden!
 - (fehlt)** Gemäß Browsereinstellungen

Video 2

■ Auswahl nach Medientyp

```
1 <video controls autoplay muted poster="vorschau.jpg">
2   <source src="vid_small.mp4" media="handheld">
3   <source src="vid.mp4" media="all">
4   <a href="vid.mp4">vid.mp4 oder... herunterladen</a>
5 </video>
```

Video 3

■ Auswahl nach Dateityp und Codec

```
1 <video controls autoplay muted poster="vorschau.jpg">
2   <source src="vid.mp4" type="video/mp4">
3   <source src="vid.ogv"
4           type="video/ogg; codecs='theora, vorbis'">
5   <a href="vid.mp4">vid.mp4 oder... herunterladen</a>
6 </video>
```

Media API

- \leadsto `HTMLMediaElement`
 - Methoden: `load`, `play`, `canPlayType`, `pause`
 - Attribute: `volume` (`[0,1]`), `muted`, `ended`,... (siehe Doku)
- Unterschied von Video zu Audio:
 - `width`, `height`, `poster`
 - `videoWidth`, `videoHeight`, kein Konstruktor
- \leadsto <http://www.jplayer.org>
 - jQuery-Multimedia-Player
 - mit HTML und CSS beliebig gestaltbar
 - einfach in der Handhabung

Codecs (coder-decoder)

Tausende Codecs!

MP3 (MPEG-1 Audio Layer 3) Audio, nicht frei

AAC Advanced Audio Encoding, nicht frei, bessere Qualität als MP3

Vorbis Audio, frei

Opus Internetstandard der IETF (seit 9/2012, RFC 6716), Audio, frei, "beste" Qualität, dzt. Firefox)

Theora Video, frei

H.264 (MPEG-4 part 10, MPEG-4 AVC) Audio/Video, nicht frei, Qualität!

VP8 Video, frei, viel besser als Theora, etwas schlechter als H.264

Konvertieren \leadsto <http://www.mirovideoconverter.com/>

Audio- und Videocontainer 1

enthalten Daten mehrerer Codecs ¹

WAV Audio

- `audio/wav, .wav`

Ogg Audio, Video, Text

- Vorteile: frei, streaming-fähig
- Audio: `audio/ogg: .ogg, .oga`
- Video: `video/ogg: .ogv`
- verwendet meist: Theora, Vorbis

MP4 Audio, Video, Text

- Vorteile: streaming-fähig
- Audio: `.m4a`, Video: `video/mp4: .mp4`
- verwendet meist: H.264

AVI Audio/Video (von Microsoft)

- Vorteile: weite Verbreitung
- Nachteile: Technik, keine Menüs,...
- verschiedene Codecs, `.avi`

¹http://en.wikipedia.org/wiki/Comparison_of_container_formats

Audio- und Videocontainer 2

Matroska Audio, Video

- Vorteile: frei, verschiedene Codecs
- `video/x-matroska`, `audio/x-matroska`
- `.mkv`, `.mka`, `.mks` (nur Untertitel)

WebM Audio, Video, basiert auf Matroska (von Google)

- Vorteile: frei, verwendet: VP8 und Vorbis
- `audio/webm`, `video/webm`

Flash Video Audio, Video, Adobe

- Vorteile: Verbreitung durch Flash Player
- verwendet: Sorenson, VP6, H.264
- 2 Formate: `.flv`, `.f4v`
- `video/x-flv`, weitere je Codec

QuickTime Audio, Video, (von Apple)

- Basis für MPEG-4, verwendet: H.264
- `video/quicktime`, `.mov`, `.qt`

Codec-Problematik

- Patente und Patentsituation, Lizenzen, Unterstützung!
- Apple: Safari nutzt Quicktime \leadsto H.264 (proprietär), MP3, WAV, manuelle Installation: WebM
- Google: H.264 (soll aus Chrome entfernt werden), Theora/Vorbis, WebM, MP3, WAV
- Mozilla: Theora/Vorbis, WebM, WAV
- Opera: Theora/Vorbis, WebM, WAV
- IE9: H.264, MP3, manuelle Installation oder Plugin: WebM

Überblick

- Neue Varianten von Eingabefeldern
- Platzhaltertexte
- Autovervollständigung
 - Vorschläge angeben
- Autofokus
- Eingabfelder können außerhalb von `form` sein
- Mehrfacheingabe für alle Eingabefelder
- “3.Zustand” für Checkboxes
- Überschreibbare Formularattribute
- Validierung
 - client-seitig vs. server-seitig!
- lediglich Opera unterstützt vollständig!!!
 - \leadsto Graceful Degradation

Formulare

```
1  <form action="/register" method="post "  
2      enctype="multipart/form-data" id="registerform">  
3      <fieldset>  
4          <legend>zur Person</legend>  
5          <label>Benutzerdaten:  
6              <input name="userdatafile" type="file"  
7                  accept="text/*" autofocus>  
8          </label>  
9          <label><input type="checkbox" name="agreed">  
10             Einverstanden  
11         </label>  
12     </fieldset>  
13 </form>  
  
14  
15 <input type="hidden" name="options" value="x,y"  
16     form="registerform">
```

Eingabefelder 1 - “alt”

- `<input type="text">` ... einzeilige Eingabe
- `<input type="password">`
- `<input type="hidden">`
- `<input type="checkbox">`
 - `cb.checked = true; cb.indeterminate = true`
→ 3. **sichtbarer** Zustand: man sieht nicht, dass markiert oder nicht markiert
- `<input type="radio">` ... name mit gleichem Wert
- `<input type="file">`
- `<input type="image">`
- `<input type="submit">` ... schickt ab
- `<input type="reset">` ... setzt Formular zurück
- `<input type="button">` ... “normaler” Button

Eingabefelder 2 - Wichtige Attribute

maxlength max. Länge der eingebaren Zeichen

size Anzahl der sichtbaren Zeichen

required Eingabe muss getätigt werden

multiple Mehrfache Eingaben möglich, durch `=,=` getrennt

pattern regulärer Ausdruck, der zutreffen muss

`... pattern="[a-z]{5,}" title="Min..." required>`

list Angabe des `datalist` Elementes

placeholder Platzhalter

value Wert

autofocus Feld bekommt Fokus

autocomplete Autovervollständigung (Vorschläge \leadsto `list`)

name eindeutiger Name innerhalb des Formulars

disabled Control ohne Funktion

readonly mittels Eingabe nicht veränderbar

novalidate keine Validierung (auch bei `<form>`)

■ z.B. um Formulardaten zwischenzuspeichern

Eingabefelder 2

- `type="search"` ... Anzeige wie Suchfelder des OS
 - sonst wie `type="text"`
 - kann "normal" nicht mittels CSS verändert werden
 - außer mittels CSS3 (`-webkit-`) `appearance` auf `none`
- `type="tel"`, `type="url"`, `type="email"`
 - Änderung des Tastenfeldes am Handy!
 - sonst wie `type="text"`
 - aber Validierung bei "url" bzw. "email"
- `type="color"` ... Farbauswahl
 - `value`, String mit z.B.: `#000000`

Eingabefelder 3

- `type="number"` ... Zahlen per Tastatur
 - meist zwei Kontrollelemente zum Inkr. & Dekr.
 - per Tastatur alles möglich
 - nicht mittels Kontrollelemente
 - nicht mittels JavaScript-API
 - Änderung des Tastenfeldes am Handy!
 - \leadsto Validierung
 - Methoden: `stepUp`, `stepDown`
 - Attribute: `valueAsNumber`, `min`, `max`, `step`
 - Defaultwerte: `step = 1`

```
1 <input type="number" min="0" max="3" step="1">
```

- `type="range"` ... wie "number", aber
 - Anzeige mittels "slider"
 - Defaultwerte: `min = 0`, `max = 100`, `step = 1`

Eingabefelder 4

- `type="date"` ... Datum (YYYY-MM-DD, MM-DD)
 - Anzeige und Eingabe gemäß Lokalisierung!
 - aber: noch nicht ausgereift: Style und Anzeige nicht veränderbar

Die folgenden werden nur von Opera unterstützt:

- `type="time"` ... Zeiteingabe (HH:MM, HH:MM:SS, HH:MM:SS.F{1,3})
- `type="datetime"` ... Datum und Zeit
 - `<Date><Blank_or_T><Time><Timezone>`
 - z.B.: 2012-09-08T10:03+02 oder 2012-09-08 08:03Z (UTC)
- `type="datetime-local"` ... Datum und Zeit ohne Zeitzone
- `type="month"` ... Monat (YYYY-MM)
- `type="week"` ... Woche, z.B. 2012-W01

DOM-Attribute

- `valueAsNumber`, `valueAsDate`

Weitere Formularelemente - “alt”

- `textarea` ... mehrzeilige Eingabe
- `button` ... ähnlich einem “input” - Button, aber mit Inhalt!
- `select`, `option`, `optgroup`

```
1 <select name="toppings" size="5" multiple>
2   <optgroup label="Gruppe 1">
3     <option>Schoko</option>
4     <option>Haselnuss</option>
5     <option>Pistazien</option>
6   </optgroup>
7   <optgroup label="Gruppe 2">
8     <option>Schlagobers</option>
9     <option>Rum</option>
10  </optgroup>
11 </select>
```

Weitere Formularelemente

- `keygen` ... Generierung von Schlüsselpaaren
- `object` ... Einbindung externer Inhalte
 - Flash, Applets, Bilder, HTML Dokumente
 - Formular versucht "Daten" daraus zu beziehen
 - wenn "plugin"
- `progress, meter`
- `output` ... Ergebnis einer "Berechnung"
 - Zugriff via `value`
 - Wird im Formular mitversendet

Verschiedenes

■ Platzhaltertexte \leadsto grauer Text...

```
1 <input type="text" name="log" placeholder="Benutzername">
```

■ Autovervollständigung

■ Standardmäßig eingeschalten

■ Attribut autocomplete: on oder off

■ \leadsto Browser merkt sich...

■ Will Entwickler vorgeben, dann:

```
1 <input type="text" list="colors">
2
3 <datalist id="colors">
4   <option value="red"></option>
5   <option value="green"></option>
6   <option value="blue"></option>
7 </datalist>
```

Überschreibbare Formularattribute

- Formular abschicken: `action`, `method`,... von `<form>`
- Weitere Attribute für Submit-Buttons:
 - `formaction` ... überschreibt `action`
 - `formtype` ... überschreibt `enctype`
 - `text/plain`
 - `application/x-www-form-urlencoded`
 - `multipart/form-data`
 - `formmethod` ... überschreibt `method`
 - `formnovalidate` ... überschreibt `novalidate`
 - `formtarget` ... überschreibt `target`
 - `_blank` ... in neuem Fenster oder Tab
 - `_self` ... selben Frame (keine Frames in HTML5, außer `<iframe>`!)
 - `_parent` ... im übergeordneten Frame
 - `_top` ... im gesamten Fenster (oberster Frame)

Formularvalidierung - 1

■ ausgeschlossene Elemente

- `<hidden>`, `<submit>`, `<image>`, `<keygen>`, `<reset>`,
`<object>`, `<meter>`, `<progress>`, `<output>`

■ ausgeschlossene Typen des `<input>` - Elementes

- `hidden`, `submit`, `image`, `reset`, `button`

■ Überprüfungen

- Pflichtfelder: `required`
 - Syntaxüberprüfungen, wie z.B. bei `type="email"`
 - Bedingungen, wie z.B. durch `min` oder `max`
 - Bedingungen durch reguläre Ausdrücke (`pattern`)
 - in JavaScript programmierte Bedingungen
- ## ■ Eingebaute Validierungen für: `email`, `number`, `url`

Formularvalidierung - 2

- Automatische Validierung eines Elementes beim Absenden, wenn:
 - von Validierung nicht ausgeschlossen
 - kein `disabled` Attribut
 - kein `readonly` Attribut
 - **muss**: ein `name` Attribut
 - innerhalb von `<form>` oder `form` - Attribut
- entweder
 - eingebaute Validierungen
 - selbst erstellte Regeln

Manuelle Validierung

■ DOM-Eigenschaften

- `feld.willValidate ... true`, wenn `feld` validiert wird.
- `feld.validity.valid ... true`, wenn korrekt
- `feld.checkValidity() ... wie`
`feld.validity.valid`, aber zusätzlich wird ein `invalid-Event` gefeuert, wenn `false`.
- `feld.validity.valueMissing ... true`, wenn Pflichtfeld nicht ausgefüllt.
- `feld.validity.typeMismatch ... true`, wenn Eingabetyp falsch.
- `feld.validity.patternMismatch ... \leadsto pattern`.
- `feld.validity.tooLong ... \leadsto maxLength`
- `feld.validity.rangeUnderflow ... \leadsto min`
- `feld.validity.rangeOverflow ... \leadsto max`
- `feld.validity.stepMismatch ... \leadsto step`
- `feld.validity.customError ... true`: eigener Fehler
- `feld.validationMessage ... gerade aktuelle Fehlermeldung (im oninvalid Event-Handler)`

CSS3 Pseudoklassen

`:valid,...` beziehen sich immer nur auf Elemente, die Validierung unterliegen

- `:default` ... Default-Submit-Button
- `:indeterminate` ... unbestimmte Checkboxes und Radioboxen
- `:valid`
- `:invalid`
- `:in-range` ... nicht zu hoch und nicht zu nieder
- `:out-of-range`
- `:required`
- `:optional` ... keine Pflichtfelder oder von Validierung ausgeschlossen
- `:read-only` ... `readonly` Attribut
- `:read-write` ... kein `readonly` bzw. nicht deaktiviert

Offline-Webanwendungen

- Application Cache
 - Dateien werden beim “Start” heruntergeladen
 - nicht mit dem normalen Browser-Cache verwechseln!
 - Unterstützung gut, aber... in IE voraussichtlich ab IE10
 - ~> API
- Online/Offline - Status bestimmen
- Daten lokal speichern...
 - WebStorage (auch DOMStorage genannt, Unterstützung ok)
 - IndexedDB (nur Chrome und Firefox, IE ab 10)
 - File API (IE10+?, Safari?)
 - WebSQL (kein Firefox, kein IE): veraltet!

Application Cache - Manifest - 1

- Inhalt (utf-8!)

```
CACHE MANIFEST
```

```
# die erste Zeile ist Pflicht!
```

```
# alle Pfade: absolut oder relativ zu Manifest  
img/foo.png  
js/bar.js
```

- MIME Typ: `text/cache-manifest`

- in HTML einbinden (relativ oder absolut, aber gleicher Origin)

```
1 <html manifest="cache-manifest.manifest">
```

Nur in "Wurzel"-HTML Datei notwendig!

- Cache wird neu angelegt, wenn
 - Änderung der Manifest-Datei
 - Benutzer den Cache löscht

Application Cache - Manifest - 2

■ 3 Arten von Abschnitten

- Über Cache verfügbar: CACHE MANIFEST bzw. CACHE
 - einzeln aufzählen
 - relative oder absolute Pfade
- Nur über Netzwerk zugreifbar: NETWORK
 - auch Teilpfade!
- Nicht verfügbar: FALLBACK
 - die Ersatzressourcen kommen in den Cache!

■ Ablauf

- 1 wenn in NETWORK, dann aus dem Web laden
- 2 Anderenfalls: in CACHE, dann aus Cache laden
- 3 Anderenfalls: in FALLBACK, dann
 - Versuch aus dem Web laden, anderenfalls aus Cache
- 4 Anderenfalls: * in NETWORK, dann aus Web laden

Application Cache - Manifest - Beispiel

CACHE MANIFEST

NETWORK:

foo.html
/blog/

CACHE:

index.html
bar.html

FALLBACK:

/ ersatz.html
/bilder/ /bilder/ersatz.png

Application Cache - API

■ Events...

- checking ... Laden oder Überprüfen der Manifest-Datei
- noupdate ... keine Änderung der Manifest-Datei
- downloading ... Manifest wird geladen
- progress ... Ressourcen werden geladen
- cached ... Ressourcen erstmals geladen
- updateready ... Cache aktualisiert
- obsolete ... Manifest \leadsto 404, dann Cache gelöscht
- error ... Fehler beim Download der Ressourcen

■ Objekt `window.applicationCache`

- zum Registrieren der Event-Handler
- zum Abfragen des Status `applicationCache.status`
 - 0 ... kein Cache vorhanden
 - 1 ... Cache vorhanden und aktuell
 - ...

Application Cache - Cache updaten

- Achtung: Ändern sich die Ressourcen (z.B. html Datei), dann keine Änderung am Cache!
- Aber: Änderung an der Manifest-Datei, dann schon.
 - Manifest-Datei wird **nur** überprüft HTML Datei geladen wird, die `manifest=...` hat!
- Aber: Manifest-Datei wird erst geladen, wenn Ressource schon angezeigt \leadsto alte Version der Ressource ist zu sehen!!
 - Erst neues Laden der Ressource bewirkt Anzeigen der neuen Version aus dem Cache!
- Wenn `updateready`, dann:

```
1 applicationCache.addEventListener("updateready", function(  
2     console.log("Cache aktualisiert");  
3     applicationCache.swapCache();  
4     alert("Lokaler Cache aktualisiert -> Neu laden");  
5     // Absenden blockieren!  
6     document.getElementById("submit").disabled = true;  
7 }, false); // false -> bubbling phase
```

Online oder Offline?

■ Status abfragen

- navigator.onLine
 - false ~> sicher offline
 - true ~> vermutlich online

■ Events

- online und offline

```
1 <!DOCTYPE HTML>
2 <html><head><title>Online status</title>
3 <script>
4   function updateIndicator() {
5     document.getElementById('indicator').textContent =
6       navigator.onLine ? 'online' : 'offline';
7   }
8 </script></head>
9 <body onload="updateIndicator()"
10   ononline="updateIndicator()"
11   onoffline="updateIndicator()">
12   <p>Network: <span id="indicator">(state unknown)</span>
13 </body></html>
```

WebStorage

- Key-Value
- 2 Arten
 - `sessionStorage` ... gerade laufende Browser-Session (Browser zu \leadsto Daten weg)
 - `localStorage` ... Daten beliebig lange (bis Benutzer löscht)
- Storage-Interface
 - `length` ... # der Datensätze
 - `key(long index)` ... Name des Schlüssels an der Position
 - `getItem(string key)` ... Liefert Wert
 - `setItem(string key, string data)` ... Setzt Wert
 - `removeItem(string key)` ... Löscht Datensatz
 - `clear()` ... Löscht alle Datensätze
- Datenspeicher nach Websites getrennt!

Geolocation-API

- Prinzipiell
 - Positionsbestimmung (GPS, IP, WLAN Access Points, GSM Zelle)
 - Benutzer muss zustimmen!
- Geschichtlich
 - Ursprünglich in HTML von WHATWG
 - jetzt eigener Standard (<http://w3.org/TR/geolocation-API/>)
- Zugriff über `navigator.geolocation`

Beispiel 1 - Einfach

```
1  function successCallback(pos) {
2      alert("Position: " + pos.latitude + " " +
3          pos.longitude + "\n" +
4          "Hoehe: " + pos.altitude + "\n" +
5          "95% Genauigkeit von Laenge&Breite (m): " +
6          pos.accuracy + "\n" +
7          "95% Genauigkeit von Hoehe (m): " +
8          pos.altitudeAccuracy + "\n" +
9          "Richtung (): " + pos.heading + "\n" +
10         "Geschwindigkeit (m/s): " + pos.speed);
11 }
12
13 navigator.geolocation.getCurrentPosition(successCallback);
```

Beispiel 2 - Genauer

```
1  var options = {
2      enableHighAccuracy: true, // supergenau, u.U. langsamer
3      timeout: 1000, // legt max. Wartezeit fest (ms)
4      maximumAge: 0 // max. akzeptiertes Cache-Alter (ms)
5  }
6
7  function errorCallback(err) {
8      // 0 ... unknown, 1 ... permission denied,
9      // 2 ... position unavailable, 3 ... timeout
10     alert(err.message + " " + err.code);
11 }
12
13 navigator.geolocation.getCurrentPosition(
14     successCallback, errorCallback, options);
```

Beispiel 3 - Überwachung

```
1  var timerid = navigator.geolocation.watchPosition(  
2      successCallback, errorCallback, options);  
3  
4  document.getElementById("stopButton").  
5      addEventListener("click",  
6          function() {  
7              navigator.geolocation.clearWatch(timerid);  
8          },  
9          false);
```
