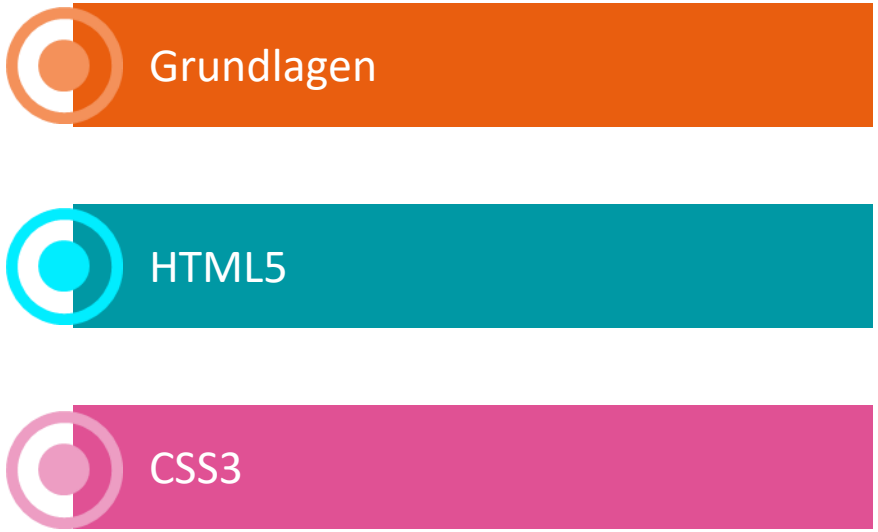
# HTML5, CSS3

HTML5, CSS3

integrata
Cegos Group

inspire
qualify
change

# Inhalts-verzeichnis

# Grundlagen


- Einführung
- Aktueller Stand

# Einführung

# Einführung

- Beteiligte Gremien
- Stand der Dinge
- Die heutige Praxis
- Links zum Thema

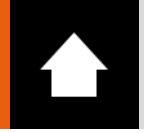
# W3C – World Wide Web Consortium

- Internationale Normungsorganisation des World Wide Web
- Gründer und Vorsitzender ist Tim Berners-Lee
- Beispiele für Technologien, die vom W3c standardisiert wurden:
  - HTML
  - XHTML
  - XML
  - CSS
  - SVG
  - …und viele mehr
- http://www.w3.org

# WHATWG – Web Hypertext Application Technology Working Group

- Arbeitsgruppe, die das Web mit Fokus auf Internetanwendungen weiterentwickeln möchte
- Wurde 2004 von verschiedenen Unternehmen gegründet, denen die Entwicklung durch das W3C zu langsam ging
- Das WHATWG ersetzt nicht das W3C, sondern entwickelt eigenständig neue Technologien, die dann dem W3C zur Zustimmung, Weiterentwicklung oder Ablehnung vorgelegt werden.
- Aktuelle Spezifikationsentwürfe
  - HTML5
  - Web Forms 2.0
  - Web Controls 1.0

**www.whatwg.org**









# Aktueller Stand

# Stand der Dinge (01/2013)

- Aktuelle, gültige Standards
  - HTML4.01 – Standard seit Dezember 1999
  - XHTML 1.0 – Standard seit 2001
  - CSS2 – Standard seit Mai1998
  - JavaScript  1.8 / ECMAScript 5.1
- Die künftigen Standards
  - CSS3 – wird entwickelt seit 2000, teilweise spezifiziert, Teile noch Entwurfsstatus
  - HTML5 – wird entwickelt seit 2004, Funktionsumfang seit 12/2012 festgelegt, offizielle Verabschiedung für 2014 vorgesehen
    - Arbeitsentwurf HTML 5.1 seit 12/2012
  - ECMAScript 5.1 seit 2011 spezifiziert

# Was geht heute?

- Der Funktionsumfang von HTML5 ist seit Ende 2012 fixiert, es besteht Klarheit für Entwickler und Browserhersteller
- Mindestumfang von CSS3 ist ebenfalls klar umrissen, teilweise auch schon verabschiedet – auch hier besteht Sicherheit
- ECMAScript 5.1 ist seit 2011 verabschiedet und ISO-standardisiert
- Zahlreiche Funktionen werden in modernen Browsern bereits heute unterstützt

ABER:

- Heute noch verbreitete ältere Browser unterstützen die neuen Standards nicht
- Kein einziger Browser unterstützt bis heute sämtliche HTML5/CSS3/JavaScript-Funktionalitäten
- Selbst „unterstützte“ Befehle werden durchaus unterschiedlich interpretiert, und müssen unter Umständen mit „Browserweichen“ abgefangen werden.

# Die heutige Praxis

- Wir befinden uns in einem Zwischenstadium in dem es gilt gute Kompromisslösungen zu finden
- Zielkonflikt zwischen Nutzung sinnvoller neuer Funktionen und mangelnder Unterstützung durch (ältere) Browser oder Plattformen
- Erhöhter Aufwand bei Entwicklung, Test und Wartung von Webapplikationen
- Vorgehensweise
  - Nutzung von HTML5/CSS3 als Nice-to-Have-Funktionen
  - Fallback-Lösungen / Workarounds für ältere Browser
  - Spezielle Seiten für HTML5-fähige (z.B. mobile) Clients

# Links zum Thema

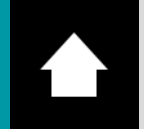
- Die beteiligten Gremien / Standardisierungsorganisationen
  http://www.w3.org
  http://www.whatwg.org
  http://www.ecma-international.org/

- HTML5/CSS3 bei W3Schools.com (englisch)
  http://www.w3schools.com

- Welcher Browser unterstützt was?
  http://caniuse.com

# HTML5


- Entstehung
- Änderungen zum klassischen HTML
- Strukturelemente
- Semantische Elemente
- Multimedia
- Formulare
- Ergänzende Steuerungselemente
- Neue Elemente für Formulare
- Accessibility



# Entstehung

# HTML5

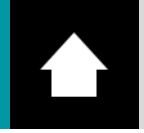
- Entstehungsgeschichte: HTML5 und XHTML2.0
- HTML5-Elemente
  - Nicht unterstützte HTML4-Elemente in HTML5
  - Neue Elemente in HTML5
  - Video-Element / JavaScript API
  - Canvas-Element
  - Web Forms 2.0

# Entstehungsgeschichte

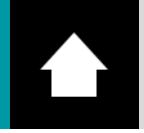
- Entstehungsgeschichte: HTML5 und XHTML2.0
  - Ziel ist die Trennung von Dokumentenstruktur, Gestaltung und Logik.
  - In HTML bis Version 4 einschließlich ist diese Trennung immer mehr verwässert worden, da die Sprache zunehmend gestalterische Aufgaben übernehmen musste und entsprechenden Anforderungen gerecht werden sollte.

# Entstehungsgeschichte Überblick

- 2003 Erste Basics
- 2004 Übernahme durch WHATWG
- 2008 Übernahme und erster öffentlicher Arbeitsentwurf
- 2012 (Dezember): Funktionsumfang festgeschrieben
- 2014 (geplant): Offizielle Verabschiedung von HTML5



# Änderungen zum klassischen HTML



# Spezifikations-änderungen: Neue Elemente in HTML5

# Spezifikationsänderungen
# Nicht unterstützte HTML4-Elemente in HTML5

# Spezifikationsänderungen
# Nicht unterstützte HTML4-Elemente in HTML5

- HTML-Auszeichnungen, die primär gestalterische Aufgaben hatten, wurden aus dem Sprachumfang entfernt, da diese Einsatzbereiche viel besser in CSS umgesetzt werden können.

  <basefont>, <big>, , <font>, <s>, <strike>, <tt>, <u>, <xmp>

- Elemente, die bereits durch andere Elemente gegeben waren und somit doppelt vorlagen, sind ebenfalls entfernt worden.
- Dazu zählen:

| HTML4 | HTML5 |
|---|---|
| <acronym> | <abbr> |
| <applet> | <object> |
| <dir> | <ul> |

# Entfernte HTML4-Attribute in HTML5

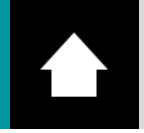
- Die nachfolgenden Attribute werden in HTML5 voraussichtlich nicht mehr unterstützt, da viele von ihnen ausschließlich gestalterische Aufgaben hatte. Durch die Trennung von Dokumentenbeschreibung, Layout und Content, gehören diese Attribute nun zum Aufgabenbereich von CSS.
- Entfernte Attribute (Auszug)
  - **abbr** und axis auf td und th
  - **align** auf caption, col, colgroup, div, h1, h2, h3, h4, h5, h6, hr, iframe, img, input, legend, object, p, table, tbody, td, tfoot, th, thead und tr
  - **alink, link, text** und **vlink** auf body
  - **archive, classid, codebase, codetype, declare und standby** auf object
  - **background** auf body
  - **bgcolor** auf body, table, td, th, und tr
  - **border** auf object und table
  - **cellpadding** und **cellspacing** auf table
  - **char** und **charoff** auf col, colgroup, tbody, td, tfoot, th, thead und tr
  - **charset** und **rev** auf a und link
  - u. v. m.

# Die neue Doctype-Definition

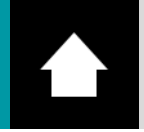
- Der Doctype muss die erste Angabe in einem HTML5-Dokument sein. Es steht also noch vor der <html>-Auszeichnung. In HTML5 wird der Doctype lediglich durch die Angabe <!DOCTYPE html> vorgenommen.

```
<!DOCTYPE html>
```

- Es handelt sich bei der Deklaration nicht um ein HTML-Tag.!
- Keine Basierung auf SGML, eigene Parseregeln, daher kein DTD-Verweis
- Gab es in HTML 4.01 noch drei unterschiedliche Doctypes, so gibt es in HTML5 nur noch den einen <!DOCTYPE html>.



# Strukturelemente



# Neue Strukturelemente

# Neue Strukturelemente

- <article>
  - Das Tag <article> sollte für einen unabhängigen Teil eines Dokumentes, einer Seite oder einer Applikation verwendet werden.
  - Zum Beispiel:
    - Forum
    - Magazin- oder Zeitungsartikel
    - Blog-Eintrag
    - Leser-Kommentar
    - Widget
    - ...

# Neue Strukturelemente

- <aside>
  - Definiert in HTML5 Dateien Inhalt, der zum Beispiel Marginalien, Definitionen oder Erläuterungen, Querverweise und zusätzliche Informationen beinhalten kann.
  - Sinnvoll ist der Einsatz des <aside> Elements bei Sidebars und auch für inhaltliche Einschübe zum Hauptbeitrag.
  - Zum Beispiel innerhalb eines <article>-Elements: Eine Sektion, die mit dem Inhalt in Beziehung steht, aber nicht Teil des Artikels sein muss.
  - Außerhalb eines <article>-Elements: Eine Sektion, die inhaltlich mit der Webseite in Beziehung steht (z. B. Sidebar).

# Neue Strukturelemente

- <footer>
  - Dieses Tag ist nahezu selbsterklärend. Es beschreibt einen Footer-Bereich unterhalb eines anderen Bereiches, also nicht zwingend die Webseite selbst.
  - Er kann Informationen wie, Autorenhinweise, Links zu verwandten Artikeln, Urheberrechtsdaten, etc. beinhalten.
  - Ein Footer kann also auch einem Artikel hinzugefügt werden. Somit können auf einer Webseite mehrere Footer-Bereiche enthalten sein.

```
<article>
<footer>Letzte Aktualisierung dieses Artikels:
  14.09.2010</footer>
</article>
```

# Neue Strukturelemente

- <header>
  - Ein header-Bereich umfasst eine Gruppe einführender oder navigatorischer Hilfen. Er beinhaltet z.B. den Kopf eines Abschnitts (h1-h6 Elemente oder eine hgroup,...), eine Inhaltsangabe, ein Suchformular, das Logo, u. s. w..
  - Verwendung:

```
<header>
  <h1>…</h1>
  <h2>   </h2>
  <img src="foo.gif" alt="bar">
</header>
```

  - Anders als das <head>-Element gibt es für das Tag <header> keine verpflichtende Position. Es kann irgendwo im Body verwendet werden, also auch innerhalb eines Artikels.

# Neue Strukturelemente

- <hgroup>
  - Mit <hgroup> können mehrere h1-h6-Elemente gruppiert werden.

```
<hgroup>
  <h1>   </h1>
  <h2>   </h2>
</hgroup>
```

# Neue Strukturelemente

- <nav>
  - Ein Block mit wichtigen Navigationslinks, z. B. die Hauptnavigation, kann mit <nav> umfasst werden.

```
<nav>
   <ul>
          <li>Eintrag</li>
          <li>Weiterer Eintrag</li>
   </ul>
</nav>
```

# Neue Strukturelemente

- <section>
  - <section> steht für einen allgemeinen Bereich eines Dokuments.
  - Es gruppiert thematisch zusammenhängenden Inhalt und kann einen eigenen Kopf- und Fußbereich enthalten.
  - Es sollte dort eingesetzt werden, wo <article>, <aside> und <nav> unpassend sind.
  - Wichtig: Es ist nicht als einfaches Containerelement gedacht, dafür sollte weiterhin das -Element zum Einsatz kommen.
  - Genau wie <article> verfügt <section> über die Eigenschaft cite=”URL” zum zitieren.

# Neue Strukturelemente

- <figure>
  - Ein <figure>-Bereich steht für eine in sich geschlossene Einheit mit Inhalt, die zwar mit einem Artikel in Verbindung stehen kann, aber davon getrennt werden kann, ohne ihn dadurch inhaltlich zu beeinträchtigen.
  - Dabei kann es sich um ein Diagramm, eine Illustration oder Grafik, eine Tabelle usw. handeln.
  - Das Tag verfügt über keine Attribute.

```
<figure>
   <img src=”umsatz_2009.jpg” alt=”Umsatzzahlen 2009”
  />
   <figcaption>
     Umsatzzahlen 2009<br />
     <a href=”zahlen.csv”>Die Umsatzzahlen als CSV-
  Datei.</a>
   </figcaption>
</figure>
```

# Neue Strukturelemente

- <figcaption>
  - <figcaption> gehört zum Tag <figure> und beinhaltet eine Legende, eine Bildunterschrift, einen Untertitel oder eine Beschriftung.

# Neue Strukturelemente

- <details>
  - Beschreibt Details zu einem Dokument oder Teile daraus. Es sollte zusammen mit dem summary-Element genutzt werden.
  - Mit dem Attribut open=”open” wird gesteuert, ob die Details sichtbar oder unsichtbar sein sollen.
  - Standardmäßig ist es nicht sichtbar.

```
<details>
    <summary>HTML5</summary>
   In diesem Dokument wird intensiv auf die Neuerungen
 von
   HTML5 eingegangen.
</details>
```

# Neue Strukturelemente

- <mark>
  - Mit <mark> lassen sich Textpassagen markieren.
  - Zum Beispiel bei einer Suchfunktion, bei der die in die Suche eingegebenen Wörter in einer Ergebnisliste mit <mark> hervorgehoben werden.
  - Damit kann ein Bezug zwischen den Wörtern und der aktuellen Benutzerinteraktion hergestellt werden.
  - Der Einsatz ist für die gleichen Aufgaben wie  oder  vorgesehen.

# Neue Strukturelemente

- <meter>
  - Das Tag <meter> ist für numerische Angaben wie Messwerte gedacht.
  - Mit diesem Tag kann eine "Spanne" dargestellt werden. Wichtig ist, dass der maximale Wert dieser Spanne bekannt sein muss.
  - Einsatz zum Beispiel bei:
    - Festplattennutzung
    - Relevanz eines Suchergebnisses
    - Wahlergebnisse
    - …
  - Nicht gedacht ist es für beliebige Angaben, die man nicht klassifizieren oder in einen Bereich einordnen kann.

```
<meter min="0" max="100" value="2"
  title="millimeters">104mm</meter>
<meter min="0" max="100" value="25"></meter>
<meter>23%</meter>
```

  - Im nachfolgenden Beispiel kann anhand der Attribute eine Bewertung des values=”91” vorgenommen werden.

```
<meter value=”91”
    min=”0” max=”100”
    low=”40” high=”90”
    optimum=”100”>A+</meter>
```

# Neue Strukturelemente

- <progress>
  - Mit <progress> kann der Status einer laufenden Aktion ausgegeben werden. Beispielsweise bei einer Installation oder eines Bilduploads.

```
<progress value="10" max="100">63%</progress>
```

  - Der Fortschritt ist
    - **unbestimmt**: Es ist unklar, wie viel Arbeit noch besteht bis eine Aufgabe beendet ist. Z. B. ein Programm wartet auf Antwort vom Server. Es ist dabei unklar, wann oder ob überhaupt eine Antwort kommt.
    - **bestimmt**: ein numerischer Wert zwischen 0 und einem gegebenen Maximum.

# Neue Strukturelemente

- <summary>
  Summary ist das header-Element des <details>-Element.

```
<details>
    <summary>HTML5</summary>
   In diesem Dokument wird intensiv auf die Neuerungen
 von
   HTML5 eingegangen.
</details>
```

oder

```
 <h1>Rosen</h1>
 <p>Rosen gehören zur Gattung der
Rosengewächse.</p>

 <details>
   <summary>Verbreitungsgebiet</summary>
   Die Familie ist weltweit verbreitet, mit
Schwerpunkt
   auf der Nordhalbkugel.
 </details>
```

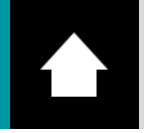
# Neue Strukturelemente

- <time>
  - Um Zeit auszuzeichnen gab es bisher keine Möglichkeit.
  - Zeitangaben waren für alle Screenreader und Suchmaschinen Bots bisher einfach Text wie alles andere auch.
  - Nun kann man Zeitangaben sinnvoll und zudem maschinenlesbar auszeichnen. Wichtige Informationen, die zeigen ob Dokumente aktuell oder veraltet sind oder ob Events abgelaufen sind oder erst noch stattfinden.

```
<time>1. August 2009</time>
<time datetime="2010-12-24">24. Dezember 2010</time>
```

# Neue Strukturelemente

- <ruby>
  - Eine ruby-Annotation sind chinesische Anmerkungen oder Zeichen. Diese werden in Ostasien verwendet, um die Aussprache des Zeichens zu zeigen.
  - Das Tag leitet einen weiteren Ruby Tag oder eine Ruby Gruppe ein. Es handelt sich um die Tags <rp> und <rt>.
    <ruby>
    <rp>
  - Mit <rp> wird ausgezeichnet was der Browser anzeigen soll, wenn er das Ruby nicht kennt.
    <rt>
    <rt> gibt die eigentliche Information wieder.



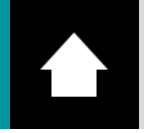



# Semantische Elemente

# Semantische Elemente

Semantisch korrekter Code hat viele Vorteile.

- Lesbarkeit/Wartbarkeit des Codes
  - Es lässt sich auf den ersten Blick erkennen, welches Element wofür steht und so der Code bei einer Änderung schneller durchschauen und ggf. ändern.
- Barrierearmut
  - Semantisch korrekter Code wird bspw. in einem Screenreader wesentlich verständlicher wiedergegeben als z. B. ein Layout als Tabelle.
- Suchmaschinenoptimierung/Maschinenlesbarkeit
  - Dadurch, dass der Code durch die korrekte Semantik maschinenlesbar wird, sind auch die Ergebnisse der Suchmaschinen besser. Durch die semantische Struktur wird auch die Struktur des Inhalts für die Suchmaschine erkennbar und kann somit die Relevanz aller Teile des Inhalts bestimmen.
- Wiederverwendbarkeit des Codes
  - Der Code kann, da man mittlerweile alleine durch CSS das Design ändern kann, bei einem Redesign leicht wieder benutzt werden.



# Multimedia

# Multimedia-Elemente

- <audio>
  - Dieses Tag dient zum Einbinden von Sound, wie Musik oder anderen Audiostreams.

```
<audio src="_data/audio.ogg" controls></audio>
oder
<audio autoplay controls>
   <source src="audio.ogg" type="audio/ogg">
   <source src="audio.mp3" type="audio/mpeg">
   <source src="audio.wav" type="audio/wav">
   <p>Ihr Browser kann die Audio-Datei leider nicht
  abspielen.</p>
</audio>
```

# Multimedia-Elemente

- <embed>
  - Mit <embed> werden, wie bisher auch schon, PlugIns wie Flash, Quicktime Movie oder Java eingebunden.

```
<embed src="helloworld.swf" />
```

# Multimedia-Elemente

- <source>
  - <source> wird in Verbindung mit dem <audio> oder <video> Element benutzt. Unglücklicherweise werden nicht alle Medienformate von allen Browsern unterstützt. Von daher ist es im Zweifelsfall nötig, eine Mediendatei in verschiedenen Formaten einzubinden.
  - Der Browser verwendet die erste Quelle, die er unterstützt.

```
<video>
   <source src="video.ogg" type="video/ogg" />
   <source src="video.mp4" type="video/mp4" />
</video>
```

# Multimedia-Elemente

- <video>
  - Mit dem neuen Tag <video> ist die Einbindung deutlich einfacher und lesbarer als in der „alten“ HTML4-Variante und funktioniert genauso wie das vom <img>-Element bekannt ist.
  - Das optionale Attribut **poster** fungiert als Titelbild und wird angezeigt, wenn das Video nicht läuft.

```
<video width="480" height="385" src="video.mp4"
  poster="poster.png">
```

<video> erlaubt es, beide Codecs als Fallback einzubinden:

```
<video>
   <source src="video.ogg" type="video/ogg"></source>
   <source src="video.mp4" type="video/mp4"></source>
   <p>Optional können hier Infos zu
  Fallbackmöglichkeiten angegeben werden.</p>
</video>
```

# Multimedia-Elemente

- <canvas>
  - Leere Fläche (Leinwand) mit einer bestimmten Größe, auf der der mit JavaScript gezeichnet werden kann (Scriptable Bitmaps).

```
<canvas width=”300” height=”150”></canvas>
```

  - Das Element ist ähnlich aufgebaut wie ein <img> Bild. Daher kann es auch wie Bilder mit den CSS-Eigenschaften margin, border, background, etc. gestylt werden.

```
<canvas width=”300” height=”150”>
<!-- Platz für Fallbackmöglichkeiten, da ältere Browser
  diese Auszeichnungen nicht kennen. -->
<img src=”alternative_zu_canvas.gif” width=”300”
  height=”150”>
</canvas>
```

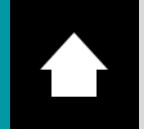
# Multimedia-Elemente

- <svg> in HTML

  Innerhalb von HTML kann auch mit Scalable Vector Graphics (SVG, W3C-Standard) gearbeitet werden.

  Vektorgrafiken können problemlos integriert und verlustfrei skaliert werden.

```
<!DOCTYPE html>
<html>
<head><title>Ein erstes SVG-Beispiel</title></head>
<body>
<!-- SVG-Insel in HTML5-Dokument -->
<svg xmlns="http://www.w3.org/2000/svg" version="1.1"
  height="100" width="200" style="border:1px solid
  #cccccc">
<circle cx="100" cy="50" r="50" style="fill:#ff0000">
</svg>
</body>
</html>
```



# Formulare

# Formulare in HTML5

- Neue <input>-Typen

# input type=„email“

- <input type=„email“>
  - Der Wert des Elements vom Typ E-Mail wird automatisch validiert, wenn das Formular abgeschickt wird.

```
E-mail: <input type="email" name="user_email" />
```

# input type=„url“

- <input type=„url“>
  - Geplant ist, dass der Wert des Elements vom Typ url automatisch validiert wird sobald das Formular abgeschickt wird.
  - `Homepage: <input type="url" name="user_url" value="" />`

# input type=„number“

- <input type=„number“>
  - Der number-Typ ist für die Eingabe numerischer Werte.
  - Dabei können Vorgaben gemacht werden, welche Werte akzeptiert werden.
  - Beispiel:
  - `Bewertung: <input type="number" name="bewertung" min="1" max="6" step=”1” />`

# input type=„range“

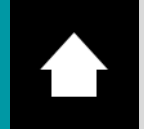
- <input type=„range“>
  - Der range-Typ kann für Eingabefelder eingesetzt werden, in denen ein Wert innerhalb eines Ranges (z.B. 1 - 10) ausgewählt werden soll. Die Darstellung erfolgt als Slider.

```
<input type="range" name="points" min="1" max="10" />
```

# input type=„date“

- <input type=„date“>
  - Eine sehr praktische Erweiterung in HTML5 ist der neue input-Typ "Date Picker".
    Er ist besonders hilfreich wenn eine Datumseingabe, z.B. für einen Flug oder ein Konzertticket, verlangt wird.
  - Die Eingabe eines Datums ist immer ein bisschen problematisch, da sie sehr häufig von dem eingegebenen Format abhängig ist, welches es dann zu validieren gilt (DD-MM-YYYY, MM-DD-YYYY, DD-MMM-YY, u. v. m.).
  - Entwickler programmieren dann Date-Picker-Widgets, die sich aber in ihrer Erscheinung sehr voneinander unterscheiden.
  - <input type=date> löst dieses Problem.

  - <input type="date">
  - <input type="month">
  - <input type="week">
  - <input type="time">
  - <input type="datetime">
  - <input type="datetime-local">



# Ergänzende Steuerungselemente

# Ergänzende Steuerungselemente

- Formularelemente oder Elemente, die dem User eine Eingabe ermöglichen oder eine Ausgabe darstellen.
  - <command>
  - <datalist>
  - <keygen>
  - <output>

# Ergänzende Steuerungselemente

- <command>
  - Ein <command> ist ein Kommando, das der Nutzer aufrufen kann.
  - Der neue Tag <command> wird zum Einbinden Formularelementen innerhalb des Tags <menu> genutzt. Formularelemente sind Button, Radio, Checkbox und Command.

```
<menu>
  <command type="command" label="Befehl">Der
 Befehl</command>
</menu>
```

# Ergänzende Steuerungselemente

- <datalist>
  - Damit kann man z. B. Listen von vordefinierten Eingaben zur Verfügung stellen. Man denke da z.B. an "Autocomplete" Funktionen in Form Elementen.

```
<input list="cars" />
  <datalist id="cars">
  <option value="BMW">
  <option value="Ford">
  <option value="Volvo">
</datalist>
```

# Ergänzende Steuerungselemente

- <keygen>
  - Mit dem HTML5-Element <keygen> können nun im Browser asymmetrische Schlüsselpaare generiert werden. Der öffentliche Schlüssel wird versendet.

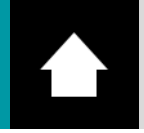
```
<form action="ziel.html" method="post"
  enctype="multipart/form-data">
   <fieldset>
   <legend>Schlüsselgenerator</legend>
   <p><keygen name="key"><input type="submit" value="Sende
  öffentlichen Schlüssel"></p>
   </fieldset>
</form>
```

# Ergänzende Steuerungselemente

- <output>
  - Das <output> Tag ist nahezu selbsterklärend. Es definiert als Gegenstück zu <input> einen Output, der bspw. durch ein Skript als Ergebnis einer Rechnung ausgegeben wird.

```
<form action="ziel.html" method="post">
<output name="sum"></output>
</form>

<form>
  <fieldset>
    <input name="a" type="number" value="0"> *
    <input name="b" type="number" value="0"> =
    <output name="result" onforminput="value = a.value *
 b.value">0</output>
  </fieldset>
</form>
```



# Neue Elemente für Formulare

# Neue Attribute für Formularelemente

- Neue nützliche Attribute für Formularelemente.
  - autocomplete
  - autofocus
  - list
  - min, max, step
  - novalidate
  - pattern
  - placeholder
  - required

# autocomplete

- Autocomplete
  - Mit dem Attribut "autocomplete" kann dem Browser erlaubt werden, die bislang gespeicherten Formulardaten zu berücksichtigen.

```
<input type="text" id="name" autocomplete="off">
```

  - Erlaubte Werte sind "on" und "off".
  - Da die meisten Browser standardmäßig jedoch die Formularwerte automatisch ausfüllen (default state des input-Feldes ist "on"), ist häufig lediglich der Wert "off" interessant.
  - Wenn der verwendete Browser dieses Attribut nicht unterstützt, wird es ignoriert.

# autofocus

- Autofocus
  - Das Attribut “autofocus” kann in einem einzelnen Formularelement eingesetzt werden um den den Fokus, also den Cursor, direkt nach dem Laden der Seite in dieses Feld zu setzen.

```
<input type="text" name="q" value=""
  autofocus="autofocus" />
```

  - Es verfügt über keine weiteren Werte.

# required

- Required
  - "Pflichtfelder" können durch das neue Attribute "required" gekennzeichnet werden.
  - Ist eines der Pflichtfelder beim "Senden" leer, wird eine browserspezifische Fehlermeldung ausgegeben und das Verschicken des Formulars wird abgebrochen.

```
<input type="password" id="pw1"
required="required">
```

# min, max, step

- Min, max, step
  - Mit "min", "max" und "step" kann ein Wertbereich definiert werden. Dabei bestimmt "min" den Mindestwert und "max" den Höchstwert.
  - Mit dem Attribut "step" wird die Genauigkeit bei numerischen sowie Datums- und zeitbezogenen Eingabetypen bestimmt.

```
<input type="number" name="myNumber" value="150"
  min="100" max="200" step="10">
```

  - Durch Klick auf die Pfeiltasten wird der Wert, hier 150, um jeweils 10 erhöht oder vermindert.

```
<input type="date" name="myDate" value="2006-07-30"
  min="2005-01-01" max="2007-12-31" step="5">
```

  - Jeder fünfte Tag ist selektierbar.

# Pattern (regexp)

- Pattern (regexp)
  - Durch die Verwendung des Attributes "pattern" ist es möglich, die Eingabe clientseitig durch reguläre Ausdrücke, den sogenannten "Pattern" zu validieren.
  - Im Falle einer ungültigen Eingabe wird, falls vorhanden, der Wert des "title"-Attributes ausgegeben.

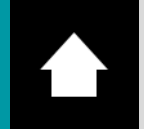
```
<input type="text" name="myRegExp" pattern="[A-Za-z]+\d+"
  title="Mehrere Buchstaben, gefolgt von mindestens einer
  Zahl">
```

  - Der reguläre Ausdruck beginnt mit einem "^" und endet mit einem "$". Es wird nicht in einem Teilstring, sondern immer das gesamte Wort verglichen. Soll innerhalb eines Teilstrings gesucht werden, muss der reguläre Ausdruck durch zwei .* umschlossen werden.

# list

- List
  - Mit dem Attribut “list” kann ein input-Element erweitert und mit datalist kombiniert werden.
  - Dadurch wird der Effekt eines frei befüllbaren Eingabefeldes erreicht. Durch die Kombination mit dem datalist-Element erhält man eine Liste mit Vorschlägen, aus die dann einer ausgewählt und übernommen werden kann.
  - Der Bezug zwischen den beiden Elementen wird durch die id=“mylist“ in datalist zu list=“mylist“ im input-Element hergestellt.

```
<label for="datalist">Suche Statistiken für…</label>
<input id="datalist" type="text" name="name"
  list="mylist" />
<datalist id="mylist">
<option value="Joe Montana" />
<option value="Jerry Rice" />
<option value="Steve Young" />
</datalist>
```



# Accessibility

# WAI ARIA

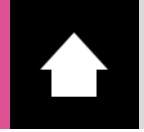
- WAI ARIA
  - WAI-ARIA steht ausgeschrieben für Web Accessibility Initiative - Accessible Rich Internet Applications.
  - Ist das Kürzel RIA bereits bekannt und im Zuge der immer populärer werdenden, webbasierten Applikationen in aller Munde, ist ARIA noch relativ unbekannt.
  - Dabei handelt es sich um eine technische Spezifikation, mit der mit JavaScript und Ajax angereicherte Webseiten und Webanwendungen besser für behinderte Menschen zugänglich gemacht werden.
  - Dies gilt insbesondere für blinde Nutzer von Screenreadern, für die die Benutzerfreundlichkeit deutlich verbessert wird und die dadurch einen besseren Zugang zu dynamischen Webseiten, bzw. den typischen Web 2.0 Applikationen erhalten.
  - Sie ist eine **rein semantische Erweiterung** für HTML, die das Layout von Webseiten nicht verändert
  - WAI-ARIA wird von den Mitgliedern der Web Accessibility Initiative (WAI), einer Gruppe innerhalb des W3C, entwickelt.

# Die 4 Säulen von WAI ARIA

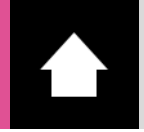
- Die 4 Säulen von WAI ARIA
  - **"Landmark Roles"** erlauben die semantische Zuweisung einer Rolle bei HTML-Konstrukten. Hiermit können Elemente, die es in HTML so eigentlich nicht gibt, für Screenreader kenntlich gemacht werden. Beispiele sind Slider, Tree Views usw.
  - **Einfache ARIA Attribute** wie "aria-required" oder "aria-invalid" lassen sich für alle HTML-Elemente verwenden und können beispielsweise dafür genutzt werden, ein Eingabefeld als ungültig zu markieren, wenn z. B. in einer E-Mail-Adresse kein @-Zeichen vorkommt, ein zweimal eingegebenes Kennwort nicht übereinstimmt usw.
  - **"Live Regions"** sind Teile einer Seite, die sich in unregelmäßigen Abständen aktualisieren. Diese Veränderungen können bei implementiertem ARIA von Screenreadern automatisch erkannt und gesprochen werden.
  - **"States und Properties"** werden für richtige JavaScript-Widgets wie beispielsweise bei einer aus Divs bestehenden Liste mit Optionen verwendet. "Activedescendant" wird z.B. benutzt, um das jeweils fokussierte Element auszuweisen, so dass in einem Popup-Menü, einer Liste usw. immer das aktive Element verfolgt werden kann. Die Tastaturnavigation muss bei eigenen JavaScript-Widgets selbst implementiert werden. ARIA ist nur eine semantische Erweiterung, die Informationen zur Verfügung stellt, selbst aber keine weiteren funktionalen Erweiterungen ermöglicht. Wenn jemand also z. B. in einer mit Divs aufgebauten Liste mit den Pfeiltasten navigieren will, muss er dies selbst implementieren und mit ARIA dann immer das fokussierte Element mitteilen. (Quelle: Wikipedia)

# CSS3


- Übersicht
- Responsive Layout
- Eigenschaften
- Webfonts
- Animationen

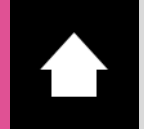


# Übersicht



# CSS3

- Exkursion: Webstandards
- CSS3 Selektoren
- Media Queries
- CSS3 Eigenschaften
- Webfonts
- Transformationen
- 3D-Transformationen
- Transitionen und Keyframe-Animation
- Übungen



# CSS3

# Exkursion: Webstandards

- Bislang befindet sich CSS3 noch in der Entwicklung. Trotzdem werden bereits heute viele Angaben von den aktuellen Browsern unterstützt.
- Um darauf hinzuweisen, dass es sich bei mancher Einstellung bislang um einen Vorschlag und noch  nicht um eine finale Funktion bzw. Standard handelt, wird von den verschiedenen Herstellern und Entwicklern ein eigenes Präfix vor die Funktion gesetzt (z. B. -webkit-border-radius: 8px).

- -khtml- KHTML: Konquerer
- -webkit- WebKit: Safari, Chrome, Arora, Mobile Safari, etc.
- -moz- Gecko: Firefox, Mozilla, SeaMonkey, etc.
- -o- Presto: Opera
- -ms- Microsoft Internet Explorer

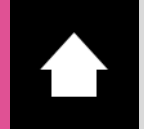
# Attribut-Selektoren in CSS3

- Selektoren ermöglichen den Zugriff auf ein bestimmtes Element
  - **E[foo^="bar"]**
    - Zugriff auf ein Element, dessen Attribut “foo” exakt mit dem Wert “bar” beginnt
      Wenn z.B. alle externen Links mit einem Icon gekennzeichnet werden sollen.

      a[href^=”http://”] { }
  - **E[foo$="bar"]**
    - Zugriff auf ein Element, dessen Attribut “foo” exakt mit dem Wert “bar” endet.
      Z.B. img[src$=”.gif”] { }
  - **E[foo*=”bar”]**
    - Element E, dessen Attribut “foo” den Wert “bar” irgendwo innerhalb des Wertes beinhaltet.
      a[title*=”wikipedia”] trifft auf alle a-Elemente zu, deren title-Attribut die Zeichenkette “wikipedia” enthält.

# Strukturelle Pseudoklassen

- Elemente werden durch ihre Struktur formatiert
- Ein bestimmtes Element wird nicht alleine nach Tag- und/oder Attributname, sondern nach Zuständen gefiltert.
- Die Formatierung kann je nach Kontext unterschiedlich sein.
- Zum Beispiel bei Zuständen von Links, alternierenden Hintergrundfarben einer Tabelle oder das erste Zeichen eines Absatzes.
- Pseudoklassen werden im CSS durch einen Doppelpunkt vom Selektor getrennt.
- Zum Beispiel (Auszug):






# Responsive Layout

# Media Queries

- Mit media queries besteht die Möglichkeit, Formatierungen für bestimmte Medien durch einen mindestens einen Ausdruck einzuschränken bzw. zu spezifizieren.
- Ein Media query besteht aus einem Medium-Typ und mindestens einem weiteren Ausdruck, der den Geltungsbereich des Medium-Typs einschränkt.
- Zum Beispiel:

```
@media screen and (max-width: 300px) {
  /* eine oder mehrere CSS Regeln */
}
```

- Bei dieser Notierung greift die Stylesheetangabe nur, wenn die Seite auf einem Bildschirm dargestellt wird und die Breite des Browserfenster kleiner als 300px ist.
- Natürlich wächst hier auch die Gefahr, mit den neuen, beinahe grenzenlosen Möglichkeiten den Überblick zu verlieren.

# Media Queries

- Bei dem weiteren Beispiel könnten unterschiedliche Styles zur Anwendung kommen, wenn ein Device, bspw. das iPad, gedreht wird:

```
body {background:#fff;}

@media all and (orientation:portrait) {
  body {background: #ff0000;}
}

@media all and (orientation:landscape) {
  body {background: #ff00ff;}
}
```

# Media Queries

- Weitere Beispiele, Eigenschaften und Einsatzbereiche (Auszug):
- Auflistung der Medium Eigenschaften:
  - width = Breite des Ausgabemediums
  - height = Höhe des Ausgabemediums
  - device-width
  - device-height
  - orientation = Ausrichtung portrait oder landscape
- @media all and (orientation:portrait)
  - aspect-ratio = Abfrage des tatsächlichen Seitenverhältnisses (z.B. des aktuellen Browserfensters)
  - device-aspect-ratio
- Seitenverhältnis des Gerätes z.B.:
- Die folgenden sind identisch, da es jeweils das gleiche Verhältnis ist.
  - @media screen and (device-aspect-ratio: 16/9)
  - @media screen and (device-aspect-ratio: 32/18)
  - @media screen and (device-aspect-ratio: 1280/720)

  - color = Abfrage, ob das Gerät Farbausgaben erlaubt
  - color-index = Abfrage, ob das Gerät eine Tabelle mit indizierten Farben verwendet
  - monochrome = Abfrage, ob das Gerät nur einfarbige Ausgaben erlaubt
  - resolution = Es kann mit min-resolution eine Mindestauflösung vorausgesetzt werden.
- @media print and (min-resolution: 300dpi)
- @media print and (min-resolution: 118dpcm) // dots per centimeter
  - scan = Abfrage, ob TV-orientierte Geräte progressive oder interlaced arbeiten

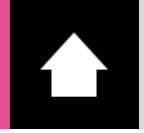
# Wo werden Media Queries im Dokument benutzt?

- HTML
  - <link rel=”stylesheet” media=”screen and (color), projection and (color)” rel=”stylesheet” href=”bsp.css”>
- XHTML
  - <link rel=”stylesheet” media=”screen and (color), projection and (color)” rel=”stylesheet” href=”bsp.css” />
- XML
  - <?xml-stylesheet media=”screen and (color), projection and (color)” rel=”stylesheet” href=”bsp.css” ?>
- CSS
  - @import: @import url(bsp.css) screen and (color), projection and (color)
  - @media: @media screen and (color), projection and (color) { ... }

# Responsive Design mit media queries

CSS-Layout/Formatierung abhängig von Endgerät / Bildschirmformat / Auflösung

```
<style type="text/css">
@media (min-width: 950px) {
   /* breite Browserfenster */
   body {background-color:green;}
 }
 @media (min-width: 450px) and
  (max-width: 950px) {
   /* Darstellung auf Netbooks */
      body {background-color:red;}
      }
 @media (max-width: 450px) {
     /* mobile Geräte */
   body {background-color:blue;}
   }
</style>
```



# Eigenschaften

# CSS3 Eigenschaften

- Zukünftig wird es keine Verabschiedung mehr für einen einzelnen CSS-Standard mehr geben. Vielmehr werden Standards für einzelne Module verabschiedet, aus denen sich dann der CSS-Standard ergibt.

  - Borders
  - Backgrounds
  - Color
  - Text effects
  - User-interface
  - Selectors
  - Basic box model
  - Generated Content
  - Other modules

# Border-image

- Border-image
- Für alle Seiten und Ecken können nun einzelne Bilder als Rahmen definiert werden.
  - border-image (shorthand), border-top-image, border-right-image, border-bottom-image, border-left-image, border-corner-image, border-top-left-image,…
- Mit CSS3 ist es möglich, ein Vorlagebild zu “zerschneiden” und die einzelnen Teile als Rahmenelemente für einzelne Seiten und Ecken anzuwenden.

# Border-image

**Beispiel**:

- Wir möchten einem DIV-Bereich einen 15px starken Rahmen zuweisen.
  `border-width: 15px`
- Das nebenstehende Bild soll als Vorlage für einen Rahmen für ein DIV verwendet werden und hat eine Größe von 90 x 90 Pixel.
- Das Bild besteht aus einem 3 x 3 Grid, bei dem in diesem Fall alle Felder gleich groß sind. Dies muss natürlich nicht so sein.
  `url(bilddatei.ext) 30 30 30 30`
- Das Bild wird nun in 9 identisch große Einzelteile zerschnitten, die für die Bildung des Rahmen an entsprechenden Positionen verteilt werden. Dabei bilden in diesem Beispiel die dunklen Elemente die Ecken und die hellen die Verbindungselemente als Zwischenteile.
- Die Art der Formatierung der Mittelteile erfolgt über die Angabe stretch, repeat oder round.
  `url(bilddatei.ext) 30 stretch`
- stretch: Das Mittelstück wird nicht dupliziert sondern auf die benötigte Länge gedehnt.
- repeat: Das Mittelstück wird einfach dupliziert und am Ende, falls nötig, einfach abgeschnitten.
- round: Beim Rendering wird versucht, die einzelnen Elemente der Duplizierung sinnvoll zu stauchen, damit die Darstellung nicht durch einen unpassenden Schnitt unharmonisch aussieht.

# Border-color

- Grundsätzlich ist die Angabe von Rahmenfarben bereits seit langem bekannt und erlaubt. Mit der Verwendung von border-color können nun mehrere Farben für einen Rahmen angegeben werden. Damit wird ein Verlauf erzielt.
  - border-color: red blue black green;
- Zusätzlich können mit border-color die Rahmenfarben auf die verschiedenen Seiten angewendet werden.
  - border-top-color
  - border-right-color
  - border-bottom-color
  - border-left-color

# Border-radius

- Mit der CSS3 Eigenschaft border-radius können (endlich :-) abgerundete Ecken erzeugt werden.

```
border-radius: 5px;
```

- Bitte beachten Sie bei der Nutzung die Verwendung des entsprechenden Präfix für den verwendeten Browser (z. B. -moz-, ...).
- Jede einzelne Ecke kann dabei auf Wunsch anders abgerundet werden.

```
border-top-left-radius
border-top-right-radius
border-bottom-left-radius
border-bottom-right-radius
```

# Box-shadow

- Mit der neuen CSS3-Eigenschaft box-shadow können sehr einfach Schlagschatten hinzugefügt werden.

```
box-shadow: 10px 13px 5px black
-moz-box-shadow: 10px 13px 5px black
```

- Die einzelnen Angaben stehen in diesem Beispiel für:
  - x-Verschiebung: 10px
  - y-Verschiebung: 13px
  - Weichzeichnung: 5px
  - Farbe: black

# Mehrere Hintergrundbilder

- Durch Komma getrennt können in CSS3 mehrere Hintergrundbilder definiert werden.
- Das bedeutet, dass für jeden Bereich nun eine einzelne Angabe vorgenommen werden kann, wo früher alle Elemente eines Hintergrundes in einem Hintergrundbild berücksichtigt werden mussten.

```
div.example {
  height: 200px;
  width: 720px;
  padding: 150px 20px 20px 20px;
  background: url(_img/body-top.gif) top left no-
  repeat,
    url(_img/banner_fresco.jpg) 11px 11px no-repeat,
    url(_img/body-bottom.gif) bottom left no-repeat,
    url(_img/body-middle.gif) left repeat-y;
}
```

# Background-size

- Hintergrundbilder müssen nicht mehr mit einem Bildbearbeitungsprogramm skaliert werden, damit sie in das Element passt.
- Für diese Formatierung gibt es in CSS3 nun die Eigenschaft background-size, die diese Aufgabe übernimmt.

```
background-size: 345px 256px;
```

# Farbangaben

- Ab CSS3 werden Farbangaben in folgenden Schreibweisen gültig sein
  - **Farbnamen wie red, blue, …**
  - **#RRGGBB**
  - **#RGB**
  - **rgb(R,G,B)**
  - **rgba(R,G,B,alpha)**
  - **hsl(hue-saturation-lightness)**
    - Der Vorteil der HSL-Schreibweise gegenüber der RGB-Schreibweise ist, dass es z. B. einfacher ist, mehrere Farben zu definieren, die den gleichen Farbton (hue) haben, sich aber durch unterschiedliche Helligkeit (lightness) und Sättigungen (saturation) unterscheiden sollen.
    - Der Farbton (hue) wird durch einen Winkel (ohne Einheit) angegeben und bezieht sich
      auf den Farbkreis.
  - **hsla(hue, saturation, lightness, alpha)**

# opacity: Deckkraft von Elementen

- Die CSS3 Eigenschaft opacity (engl. Deckkraft) kann die Deckkraft von Elementen steuern.
- Dadurch ist es möglich transparente Objekte zu erzeugen.
- Transparenz für verschiedene Browser einstellen:

```
<style type="text/css">
div {
  filter:          Alpha(opacity=70);
  opacity:         0.2;
  -moz-opacity:     0.2;
  background:      #b00;
}
</style>

<div>
  Content
</div>
```

# Farbverläufe

- Unterschieden wird zwischen zwei Arten von Farbverläufen:
- Lineare Verläufe, geradliniger Farbverlauf von zwei oder mehr Farben
- Radiale Verläufe, kreisförmiger Farbverlauf zwischen zwei Farben

```
div.linear {
  width: 250px;
  height: 250px;
  background: -moz-linear-gradient(yellow, green);
}

div.radial {
  width: 250px;
  height: 250px;
  background: -moz-radial-gradient(yellow, green, black,
  white);
}
```

- Beispielwerte:
  - background: -moz-linear-gradient(top, yellow, green); // von oben nach unten
  - background: -moz-linear-gradient(bottom, yellow, green); // von unten nach oben
  - background: -moz-linear-gradient(240deg, yellow, green); // drehen
  - background: -webkit-gradient(linear, yellow, green); // Safari (Webkit); linear | radial

# Text-shadow

- Die Eigenschaft text-shadow steuert den Textschatten

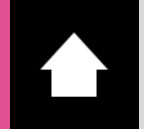
```
text-shadow: 6px 4px 10px black;
```

- Zuordnung der Werte in diesem Beispiel:
  - 6px x-Verschiebung
  - 4px y-Verschiebung
  - 10px Weichzeichnung
  - black Farbe

# Mehrspaltige Layouts

- CSS3 bietet mit der Eigenschaft “multi-column-layout” die Möglichkeit, Text in mehreren Spalten darzustellen.
  - column-count Spaltenzahl
  - column-width Spaltenbreite
  - column-gap Spaltenabstand
  - column-rule Definition einer Linie zwischen den Spalten

```
div.spalte {
 column-count: 3;
 column-gap: 15px;
 column-rule: 1px solid #aea898;
}
```






# Webfonts

# Webfonts

- In CSS3 lassen sich mit der @font-face Regel eigene Schriften im TTF-Format einbinden.
- Die nötigen Schritte im Einzelnen zuerst im Anweisungsblock der @font-face Regel:
  - **font-family**
    Der Name, mit dem die Schrift später bei den Formatierungen angesprochen werden soll. Dieser kann frei erdacht werden. Im nachfolgenden Beispiel “whoa”.
  - **Src**
    Bei src (source) kann/sollte zuerst mit der Angabe local einer oder mehrere Schriftnamen angegeben werden, unter denen die Schrift auf dem Zielsystem verfügbar sein könnte. Das können durchaus mehrere sein.

```
@font-face {
  font-family: whoa;
  src: local("Whoa regular"),
   local("Whoa-regular"),
   url(_data/whoa_regular.ttf);
}

div {
  font-family: whoa;
  font-size: 4em;
  color: firebrick;
}
```

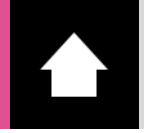
- Für Browser, die noch keine Web Fonts unterstützen gibt es auch hier die standardmäßige Fallbackmöglichkeit.
- font-family: whoa, arial, tahoma;

# Vorteile von Webfonts

- Skalierbarkeit
  - “Echte” Schriftelemente lassen sich gegenüber Bildelementen im Pixelformat (gif, jpg, png,..) verlustfrei skalieren.
- Schnellerer Download
  - Wichtig vor allen Dingen im Bereich des mobilen Web (EDGE, GPRS, UMTS, ...).
- Barrierefrei
  - Da es sich um Schrift handelt kann diese auch von Screenreadern vorgelesen oder auch von anderen Ausgabemedien verarbeitet werden.
- Suchmaschinenoptimierung
  - Die Schriftinhalte können von Suchmaschinen indexiert werden.
  - Bei Bildern konnte bislang maximal das alt- und das title-Attribut ausgewertet werden.
- Cross-Browser-kompatibel

# Integration von Webfonts

- Lokale Implementierung
  - Schriftdateien auf eigenem Server
  - Lizenzbedingungen beachten
- Implementierung über externen Server (z.B. Google Webfonts)
  - Einige Schriften lassen sich (wegen Lizenzbedingungen) nur extern einbinden
  - Sehr einfache Implementierung
  - Aber externer HTTP-Request (Datenschutz-, ggf. Verfügbarkeitsprobleme)



# Animationen

# Transformationen

- Mit der Eigenschaft transform können Elemente in CSS3 transformiert, also verschoben, skaliert, gekrümmt oder gedreht werden.
  - Scale skalieren
  - Skew krümmen
  - Rotate drehen
  - Translate verschieben

```
div {
   transform:
      skew(30deg, -10deg)
      rotate(30deg)
      translate(100px, 20px)
      scale(0.8);
}
```

  - Nebenstehendes Beispiel eines mit der transform-Eigenschaft erstellten Würfels mit Bildern und Videos auf den Seiten.

# Transform-origin

- Mit transform-origin wird der Ursprung aller Transformationen eingestellt.
- Voreinstellung ist der Mittelpunkt des zu transformierenden Objekts.

```
transform-origin: 50% 50%;
```

# Transitionen und Keyframe Animationen

- Bislang wurden in HTML4 und CSS2 Animationen meistens mit JavaScript erzeugt.
- Die CSS3-Eigenschaft Transition ermöglicht einfache "Bewegungsmuster" ohne JavaScript.
- Einfach bedeutet, das zu animierende Objekt hat einen Start- und einen Endzustand und wird ggfls. wieder vom End- zum Ausgangszustand zurückanimiert.
- Vor allen Dingen in diesem Punkt unterscheidet sich Transition von der Keyframe Animation.

| | |
|---|---|
| transition = | shorthand / eine Kurzschreibweise für die verschiedenen möglichen Eigenschaften. Alternativ hätten alle einzeln aufgeführt werden müssen. |
| transition-properties = | Welche CSS-Eigenschaften |
| transition-duration = | Animationsdauer in z.B. Sekunden |
| transition-timing-function = | Art der Animation |
| transition-delay = | Verzögerung vor dem Start |

# Beispiel Transition

- Bei dem Beispiel verschiebt sich die Position der Box beim überfahren mit der Maus innerhalb von 2 Sekunden um 100 Pixel nach rechts und anschließend wieder zurück.
- Die Notation -webkit-transition ist für webkit-basierte Browser.
- Sofern andere Browser die Eigenschaft unterstützen, ist die Funktionsweise entsprechend, es muss momentan lediglich das entsprechende Browserpräfix (-moz-, -o-, usw.) vorangestellt werden.

```
.box {
  position: absolute;
  left: 20px;
  width: 150px;
  border: 1px dotted black;
  padding: 5px;
  -webkit-transition: left 1s ease-in-out;
}
.box:hover {
  left: 120px;
}

<div class="box">
  Menüpunkt
</div>
```

# Keyframe Animationen

- Die Keyframe Animation eignet sich, wenn das zu animierende Objekt mehrere unterschiedliche Zustände (Schlüsselzustände) haben kann
- Mit @keyframes wird die Regel für einen Schlüsselzustand definiert.
- Im nachfolgenden Beispiel der Schlüsselzustand der Animation mit dem Namen NAME.
- Der Ausgangszustand wird innerhalb von from definiert - der Endzustand in to. Dazwischen können mit Prozentangaben relative Zwischenzustände definiert werden.
- Die syntaktische Definition folgt der Regel:

```
@keyframes NAME {
    from { ...
    }
    50% { ...
    }
    to { ...
    }
}
```

- Aufgerufen wir die Animation über animation-name.

```
div.animation {
     animation-name: NAME;
}
```

- Auf den div-Container wird die Animation NAME, die in @keyframes definiert wurde, aufgerufen.

# Keyframe Animationen Attribute

- animation-name: bewegen; /* Name der Animation */
- animation-duration:2s;  /* Geschwindigkeit, Dauer */
- animation-delay:1s; /* Anfangsverzögerung */
- animation-timing-function: ease-in-out; /* Animationspfade ("hereinbremsen" etc.) */
- animation-iteration-count:5; /* Anzahl Durchläufe */
- animation-direction:alternate; /* Durchlaufrichtung normal, reverse, alternate, alternate-reverse */
- animation-fill-mode:both; /* Anfangs und Endzustand sollen dargestellt
  werden */