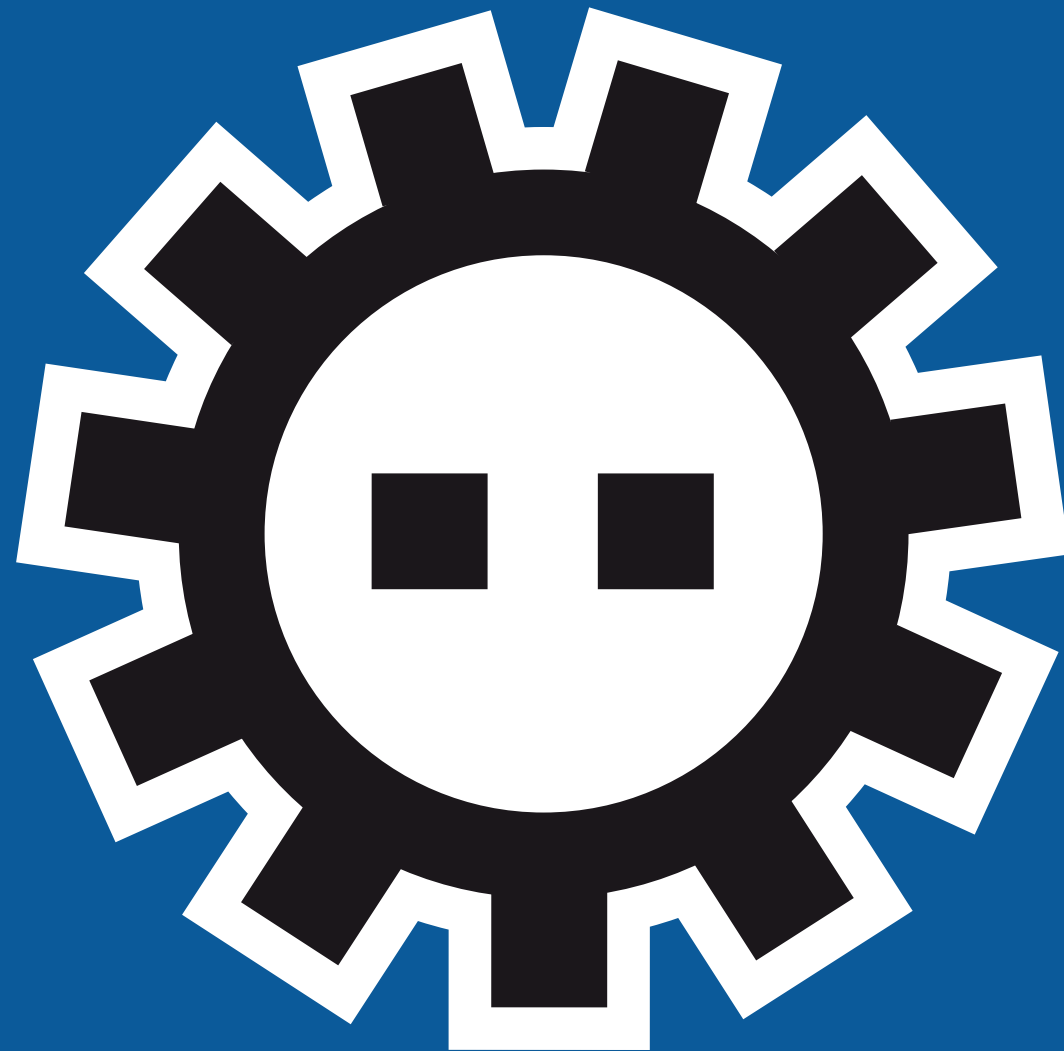


# SOFTWARE ENGINEERING 2

---

04 - HTML5



# MOTIVATION

# HTML5

- Ist in aller Munde
- Jeder hat eine eigene Vorstellung davon, was es bedeutet
- Was kann HTML5?
- Wichtiger: Welche Browser unterstützen HTML5?
- Wir betrachten die Entstehungsgeschichte

# HTML



© W3C

# HTML 1-4

- HTML (1992) - Urversion, nur Text
  - HTML 2.0 (1995) - Bilder und Formulare
  - HTML 3.2 (1997) - Tabellen, Applets
  - HTML 4.0 (1997) - Stylesheets, Skripte und Frames
  - HTML 4.0.1 (1999) - kleinere Verbesserungen
- ➡ Seit 1999 keine Änderungen mehr am HTML-Standard!

# W3C 1999 - 2009

- Was hat das W3C dann über 10 Jahre gemacht?
  - ▶ XHTML → HTML mit XML Konventionen
    - » Elemente müssen korrekt verschachtelt sein
    - » Groß-/Kleinschreibung wird unterschieden
  - ▶ XForms
    - » mächtige (aber auch extrem komplexe) Formulare
- Tendenz zur Technokratisierung
- »Empfehlungen« wurden von Web-Entwicklern und Browserherstellern nicht freudig aufgenommen

# Die Anderen

- Browserhersteller wollten abseits des starren W3C praxistaugliche Standards erschaffen
- im Unterschied zu IE und Netscape in den 90ern aber gemeinsam
- Web Hypertext Application Technology Working Group (WHATWG)
  - 2004 gegründet von Opera, Apple und Mozilla
- praktikable Erweiterungen des HTML-Standards
- Ideen wurden zusammengefasst unter dem Begriff »HTML5«

# Die Zukunft

- Das W3C arbeitete neben XHTML auch an HTML 5.0
- 2007 sah man ein, dass HTML 5.0 zu komplex und HTML5 der WHATWG praktikabler war
- seitdem arbeiten W3C und WHATWG gemeinsam an HTML5
- HTML5 wird ein fortlaufender Standard (living standard) sein
  - ▶ neue Elemente werden einfließen
  - ▶ HTML6 wird es nicht geben
- früher: Standard definiert Sprachumfang, Browser implementieren Standard
- heute: Browser implementieren neue Funktionen, Standard nimmt gute Entwicklungen auf

# www.caniuse.com

**Can I use**

## CSS

- @font-face Web fonts
- Blending of HTML/SVG elements
- calc() as CSS unit value
- 2.1 selectors
- background-blend-mode
- background-position edge offsets
- clip-path property
- Counters
- Feature Queries
- Filter Effects
- font-size-adjust
- font-stretch
- Generated content for pseudo-elements
- Gradients
- Grid Layout
- Hyphenation
- inline-block
- Masks
- min/max-width/height
- outline
- position:fixed

### # Form validation - WD

Method of setting required fields and field types without requiring JavaScript

Germany 77.51% + 4.09% = 81.6%  
Global 68.59% + 3.84% = 72.43%

Current aligned Usage relative Show all

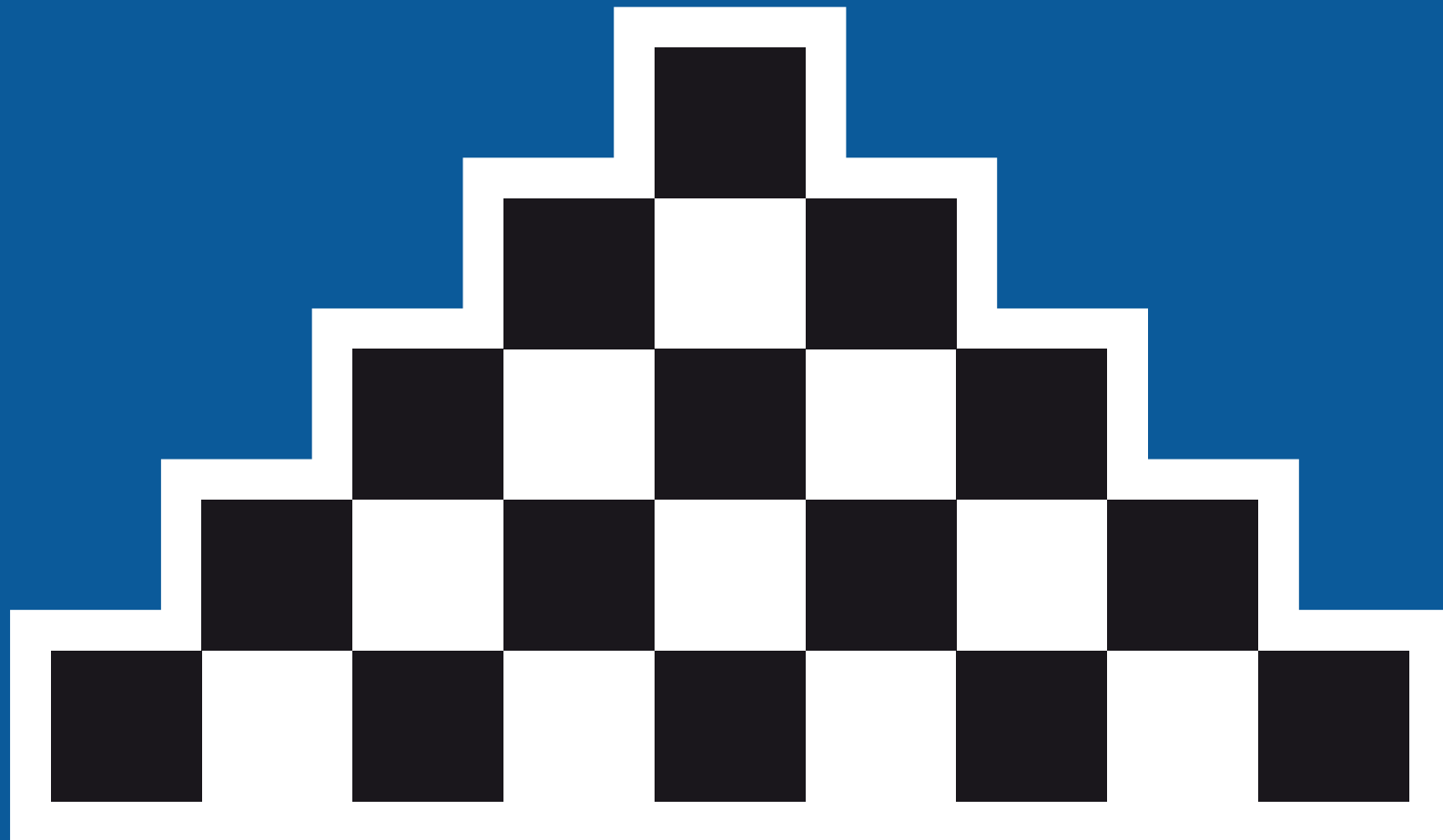
IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
							2.3	
							4	
8	30	35	5.1				4.1	
9	31	36	6.1	12.1	7.1		4.3	
10	32	37	7	24	8		4.4	
11	33	38	8	25	8.1	8	4.4.4	38
	34	39		26			37	
	35	40		27				
	36	41						

Notes Known issues (2) Resources (3) Feedback

Partial support in Safari refers to lack of notice when form with required fields is attempted to be submitted. Partial support in IE10 mobile refers to lack of warning when blocking submission.

Legend: ■ = Supported ■ = Not supported ■ = Partial support ■ = Support unknown





# GRUNDLAGEN

# Überblick

## W3C HTML5

Application  
Cache

Form Widgets

HTML/DOM  
Extensions

Semantic HTML  
Elements

## WHATWG HTML5

Canvas 2D  
Context

Web Messages

Microdata

Device Element

## »HTML5«

SVG

Geolocation

WebGL

Web Workers

Web Sockets

Web Storage

Web SQL  
Database

Selectors API

CSS3

MathML

...

# Was kann HTML5?

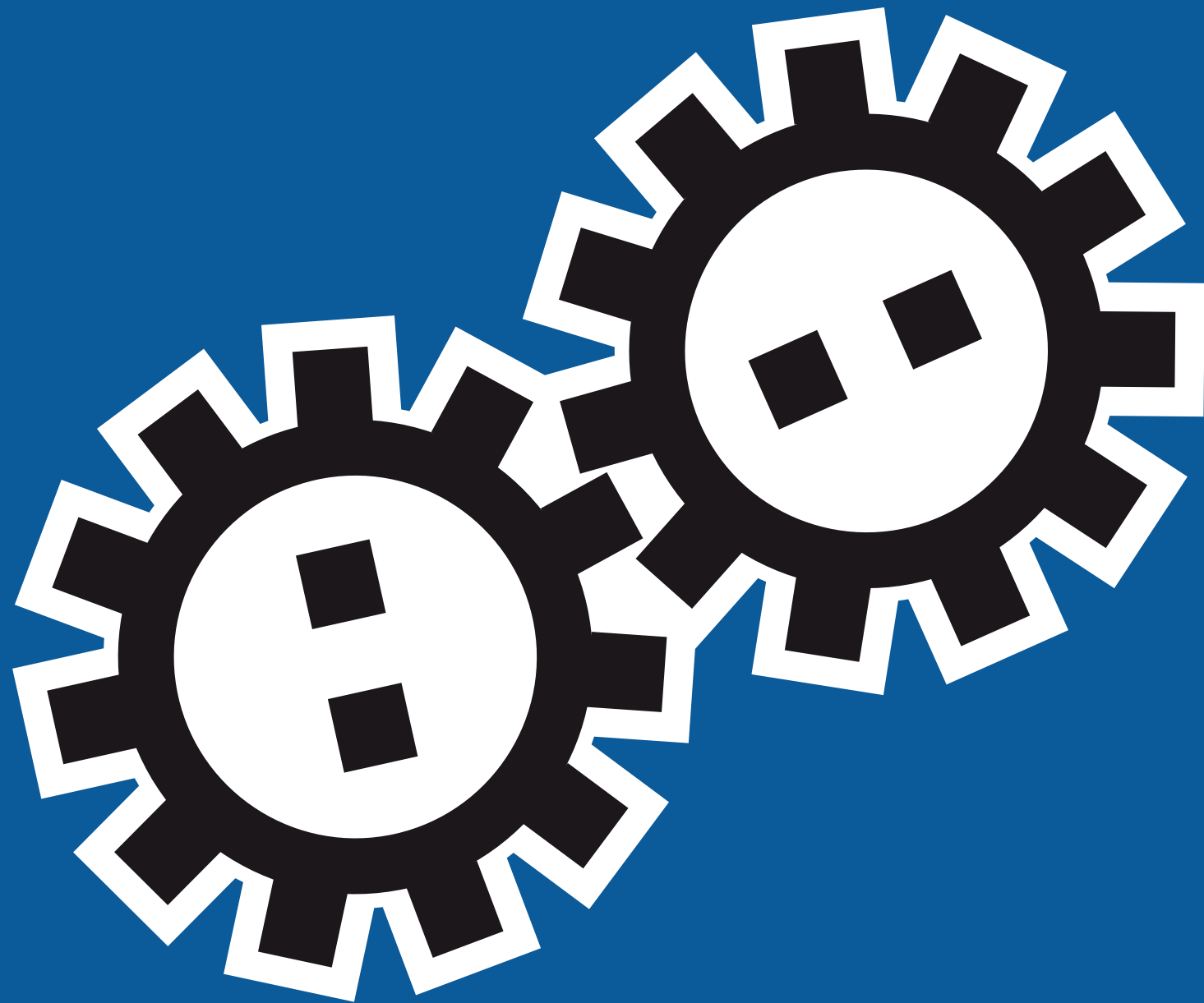
- HTML4 dient der Darstellung von Dokumenten
- HTML5 dient der Erstellung von Web-Anwendungen
  - ▶ viele Neuerungen sind nur sinnvoll einsetzbar, wenn JavaScript verwendet wird → spätere Vorlesung
  - ▶ einige Neuerungen können allerdings auch im HTML-Dokument verwendet werden → folgende Folien
- Viele Browser teilen sich den Markt → Niemand sollte ausgeschlossen werden
  - ▶ Bei jeder neuen Funktion wird Browserunterstützung genannt
  - ▶ gegebenenfalls Fallbacks bereitstellen

# Wie verwendet man HTML5?

- Die Version wird mit dem DOCTYPE des Dokuments bestimmt
- HTML 4.01 z.B.:  

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"  
    "http://www.w3.org/TR/html4/strict.dtd">
```
- HTML5 vereinfacht dies zu:  

```
<!DOCTYPE html>
```
- Kein Versions- und Grammatikangaben mehr, da fortlaufender Standard!
- Alle HTML4-Elemente werden weiterhin unterstützt
  - ▶ bis auf `<blink>`

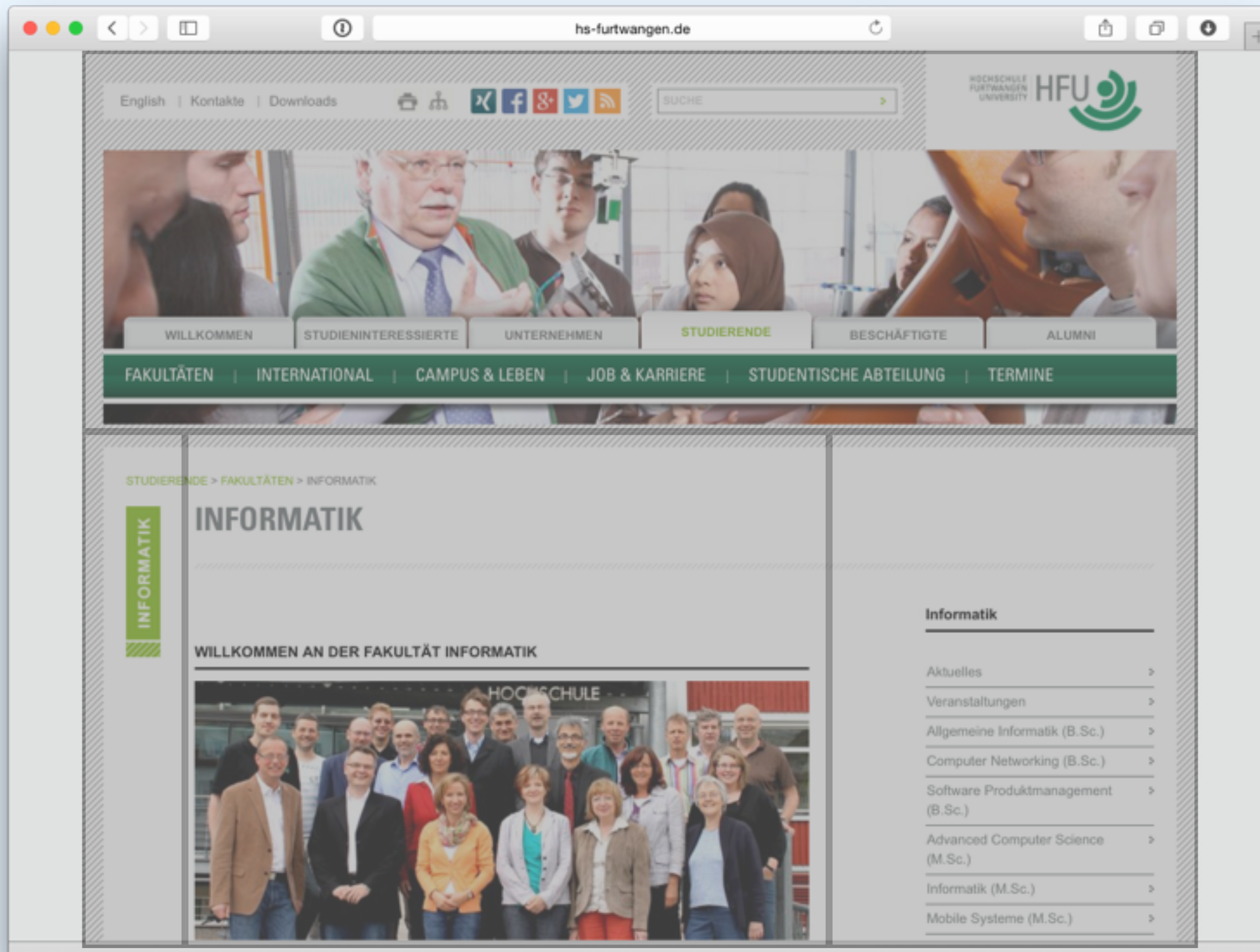


# TECHNIKEN

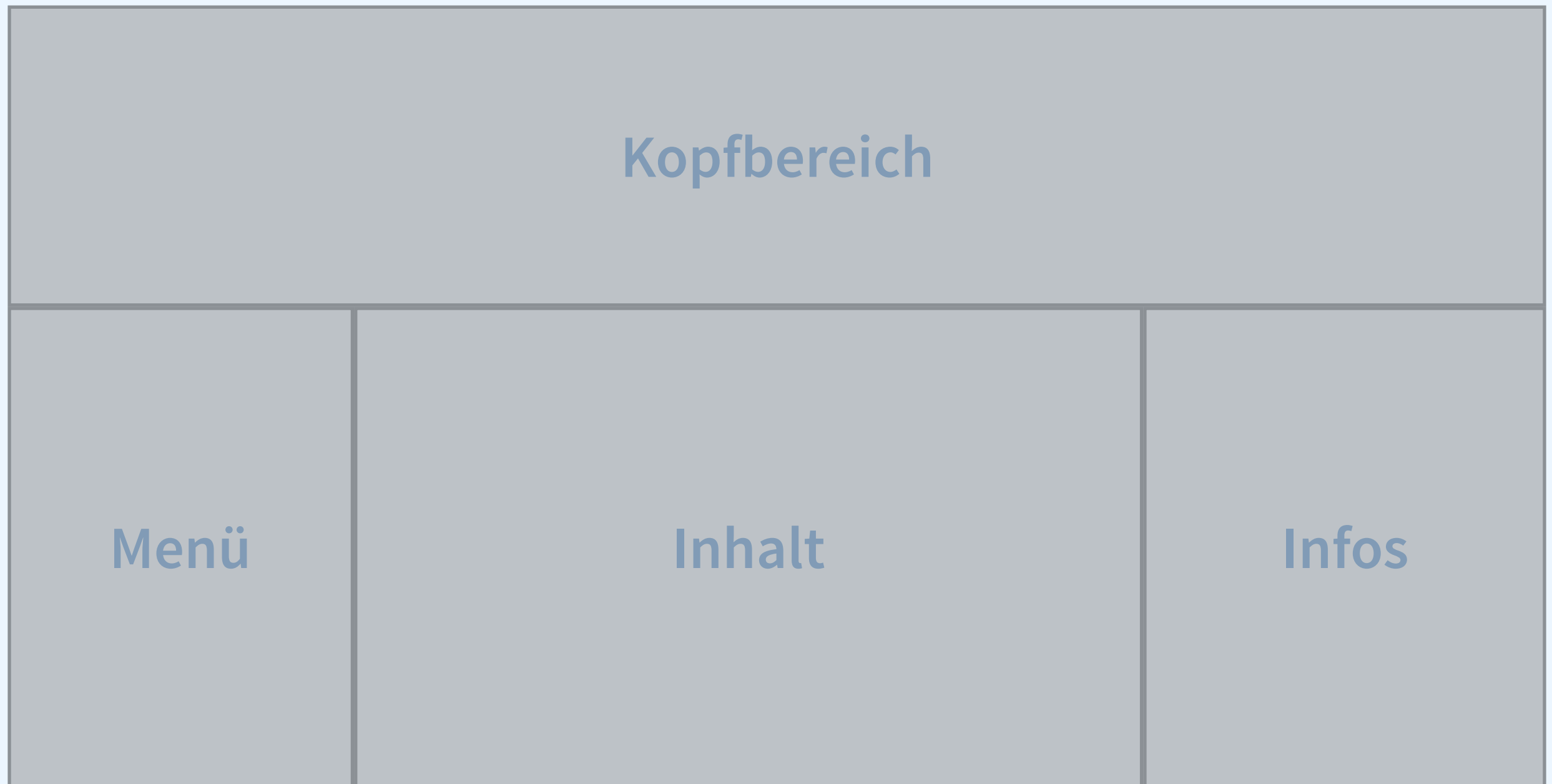
Elementen Bedeutung geben

# **semantische Elemente**

# klassische Webseite



# 3-Spalten Layout





# HTML-Dokument

```
<!DOCTYPE html>
<html>
<body>
  <div id="page">
    <div id="header">...</div>
    <div id="columns">
      <div id="menu-position">...</div>
      <div id="main-position">
        <div id="fallgrube">-</div>
        <div id="content">
          <h2>Willkommen an der ...</h2>
          <div class="article">...</div>
        </div>
      </div>
    </div>
  </div>
</body>
</html>
```



- Die Seite besteht aus div-Blöcken
- id-Attribute deuten zwar Semantik an, sind aber beliebig
- Für den Browser sind alle div-Elemente gleich
  - ▶ "divitis" oder "div-Suppe"
- Ist doch egal, das Markup sieht man als Betrachter gar nicht?

# Zielgruppen

- Wer profitiert von semantischem Markup?
- Menschen mit Behinderungen
  - ▶ barrierefreie HTML-Dokumente können von Screenreadern verarbeitet werden
  - ▶ interessiert Entscheidungsträger leider nur selten
- größter blinder Anwender der Welt: Suchmaschinen
  - ▶ elektronische Agenten, die automatisch neue Artikel indizieren
  - ▶ interessiert Entscheidungsträger brennend

# Gliederungselemente 1

- header → ein Blockelement, das den Kopfbereich enthält
  - sollte das erste Element auf der Seite sein
  - enthält Logo, Titel, Such- und Loginmasken
- footer → ein Blockelement, das den Fußbereich enthält
  - letztes Element mit rechtlichen Links und Hinweisen
- nav → ein Blockelement, das die Hauptnavigation enthält
  - sollte nicht für jede Sammlung von Links, z.B. Sponsorenlinks, verwendet werden
  - auch eine Seite mit Suchergebnissen ist kein nav-Block, sondern Inhalt

# Gliederungselemente 2

- `aside` → ein Blockelement, das die Informationen »neben« dem Inhalt enthält
  - hier können verwandte Themen, Werbung, etc. stehen
- `article` → ein Blockelement, dessen Inhalt alleinstehend sinnvoll ist
  - für den eigentlichen Inhalt der Seite
  - kann mehrmals auftauchen, auch verschachtelt (Artikel und Kommentare)
- `section` → ein Blockelement, dessen Inhalt ein Abschnitt in einem größeren Zusammenhang ist
  - üblicherweise innerhalb eines `article`

# Gliederungselemente 3

- `time` → ein Inline-Element, dessen Inhalt eine Zeitangabe darstellt  
`<time>30.10.1996</time>`
- `figure` → ein Blockelement, dessen Inhalt Darstellung enthält  
`<figure>`  
    ``  
    `<figcaption>Bildbeschriftung</figcaption>`  
`</figure>`
- und viele weitere mehr...

# HTML5-Dokument

```
<!DOCTYPE html>
<html>
<body>
  <header>...</header>
  <nav>...</nav>
  <article>
    <h1>Willkommen an der ...</h1>
    <time>30.10.1996</time>
    <figure>
      
      <figcaption>
        Hörsaal
      </figcaption>
    </figure>
    <p>...</p>
  </article>
  <article>...</article>
  <aside>...</aside>
  <footer>...</footer>
</body>
</html>
```

- div-Elemente wurden durch semantische Elemente ersetzt
- Die aktuellen Browser unterstützen diese Elemente
  - ▶ stellen sie aber nur als Blockelement dar
  - ▶ CSS ist noch notwendig

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser
								2.1	
		12.0				3.2		2.2	
	7.0	13.0				4.0-4.1		2.3	
	8.0	14.0	20.0	5.1		4.2-4.3		3.0	
Current	9.0	15.0	21.0	6.0	12.0	5.0-5.1	5.0-7.0	4.0	7.0
Near future	10.0	16.0	22.0		12.1	6.0			10.0
Farther future		17.0	23.0		12.5				

# Erweiterungen für web-basierte Formulare

# **Formularelemente**

# Exkurs: HTML 4.0 Formulare

- Formulare erlauben Interaktivität in HTML-Seiten
  - nicht nur Ausgabe, sondern auch Eingaben
- Freitextformulare
  - Kontaktseite, Kommentarfeld
- Strukturierte Formulare
  - Suchmasken, Eingabemasken
- Formulare werden mit HTML-Elementen zusammengebaut



# Formularelemente

- Formularbereich: Element »form«
  - ▶ Attribut »action« definiert Ziel-URL für die Formulardaten
  - ▶ Attribut »method« definiert HTTP-Request-Methode
    - » GET oder POST
- Zur Auswertung muss unter der Ziel-URL ein Server die Daten verarbeiten → spätere Vorlesung
- Während Browseranfragen in der Regel GET-Anfragen sind, können Formulardaten sowohl mittels GET als auch mittels POST übermittelt werden → Wo ist der Unterschied?

# Formulardaten mit GET

temperatur.jsp

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Temperaturabfrage</h1>
    <form method="get">
      ...
```

http://localhost:8080/temperatur.jsp?eingabefeld=44225

## Temperaturabfrage

Postleitzahl:

Stadt	Celsius	Fahrenheit
Dortmund	15.0	59.0

## Http-Request

### Anfrage-Zeile

GET /temperatur.jsp?eingabefeld=44225 HTTP/1.1

### Header-Zeilen

Host: localhost:8080  
User-Agent: Mozilla/5.0

### Body

<leer>

# Formulardaten mit POST

temperatur.jsp

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Temperaturabfrage</h1>
    <form method="post">
      ...
```

http://localhost:8080/temperatur.jsp

## Temperaturabfrage

Postleitzahl:

Stadt	Celsius	Fahrenheit
Dortmund	15.0	59.0

## Http-Request

### Anfrage-Zeile

POST /temperatur.jsp HTTP/1.1

### Header-Zeilen

Host: localhost:8080  
User-Agent: Mozilla/5.0

### Body

eingabefeld=44225

# Formulardaten

GET	POST
Daten werden als Parameter in der URL übermittelt.	Daten werden im Body der Anfrage übermittelt.
Daten können einfach manipuliert werden (Adresszeile des Browsers)	Daten sind schwierig (aber nicht unmöglich) zu manipulieren
Daten müssen kodiert werden. Hallo Günter → Hallo+G%FCnter	Daten können binär übertragen werden.
Länge der URL (und damit der Daten) ist beschränkt.	Daten können beliebig lang sein.
URL enthält Adresse der Webseite und Daten der Eingabe, bleiben in Bookmarks erhalten.	URL enthält nur die Adresse der Webseite, Daten werden in Bookmarks nicht gespeichert.
Wiederholtes Absenden der gleichen Daten hat keine Nebeneffekte	Wiederholtes Absenden der gleichen Daten hat Nebeneffekte und soll verhindert werden
<b>Formulardaten dienen der Abfrage von Daten (z.B. Suchmaske)</b>	<b>Formulardaten dienen der Speicherung von Daten (z.B. Bestellung, E-Mailversand, etc.)</b>

# Eingabefelder

- Element »input« zur Definition eines Eingabefeldes
- Attribut »name« zur Definition des Parameternamens
- Attribut »type« um eine Variante auszuwählen:
  - ▶ »text« → einzeiliges Eingabefeld
  - ▶ »password« → einzeiliges verdecktes Eingabefeld
  - ▶ »file« → Feld für das Hochladen von lokalen Dateien

# Eingabefelder

```
<form action="verarbeite-daten.html">
  <p>einzeilige Eingabe:<br>
  <input type="text"
        name="einzeilig"
        size="30"
        maxlength="30"/>

  </p>
  <p>einzeilige verdeckte Eingabe:<br>
  <input type="password"
        name="passwort"
        size="30"
        maxlength="30"/>

  </p>
  <p>Eingabe einer Datei:<br>
  <input type="file"
        name="datei"
        accept="text/plain"/>

  </p>
  <p>mehrzeilige Eingabe:<br>
  <textarea name="mehrzeilig"
          rows="4"
          cols="20">

  </textarea>
</p>
</form>
```


**einzeilige Eingabe:**

**einzeilige verdeckte Eingabe:**

**Eingabe einer Datei:**

  fragen.txt  
  

**mehrzeilige Eingabe:**

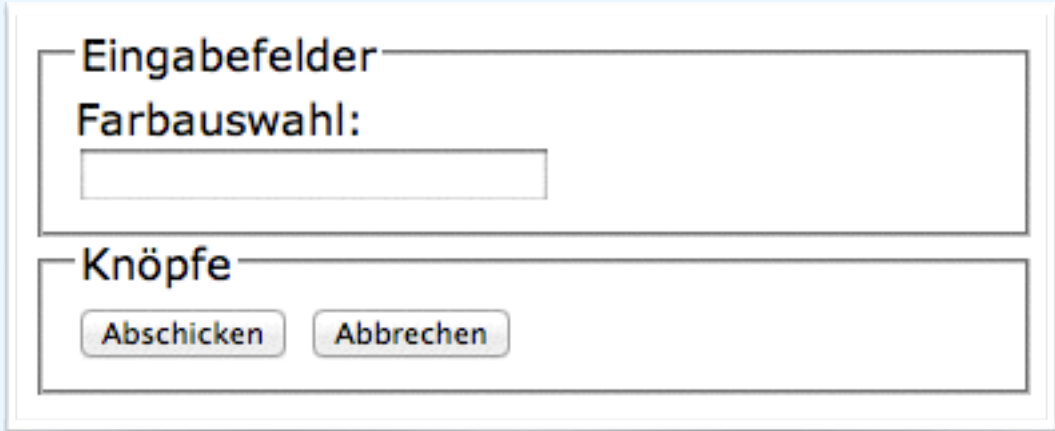
Eingabe können sich  
über mehrere Zeilen  
erstrecken. Bei Bedarf  
werden Scrollleisten

# Gruppierungen

- Zusammengehörige Eingabefelder können gruppiert werden
  - ▶ mit einem Rahmen und einer Überschrift versehen
- Element »fieldset« umschließt die Gruppe
- Element »legend« definiert die Beschriftung des Rahmens

# Beschriftungen

```
<form action="verarbeite-daten.html">
  <fieldset>
    <legend>
      Eingabefelder
    </legend>
    Farbauswahl:<br/>
    <input type="text"
          name="farbe"
          size="30" />
  </fieldset>
  <fieldset>
    <legend>
      Knöpfe
    </legend>
    <input type="submit"
          value="Abschicken"/>
    <input type="reset"
          value="Abbrechen"/>
  </fieldset>
</form>
```



The image shows a visual rendering of the HTML form code. It consists of two distinct sections, each enclosed in a rounded rectangular frame. The top section is titled 'Eingabefelder' (Input Fields) and contains the label 'Farbauswahl:' followed by a single-line text input field. The bottom section is titled 'Knöpfe' (Buttons) and contains two buttons: 'Abschicken' (Submit) and 'Abbrechen' (Reset).



# HTML5 Formulare

- Die vorgestellten Formularelemente gibt es seit 1993
  - ▶ seit 20 Jahren kaum Änderungen, neue Elemente konnten nur mit JavaScript »simuliert« werden
- HTML5 bietet Neuerungen auf zwei Ebenen
  - ▶ alte Formularelemente bekommen neue Eigenschaften mittels neuer Attribute
  - ▶ neue Formularelemente werden vorgestellt

# Verpflichtende Felder

- required-Attribut
  - das Eingabefeld ist verpflichtend
  - Formulare mit nicht ausgefüllten required-Feldern dürfen nicht abgeschickt werden
  - Browser sollten die Felder visualisieren

```
<form action="formular-verarbeiten.html">  
  Benutzername:  
  <input type="text" name="username"  
        required="required" />  
  <input type="submit" />  
</form>
```

Firefox



The screenshot shows a web form with the label "Benutzername:" followed by a text input field. To the right of the input field is a button labeled "Daten absenden". A black tooltip with white text "Bitte füllen Sie dieses Feld aus." is displayed below the input field, indicating that the field is required and must be filled out.

Opera



The screenshot shows a web form with the label "Benutzername:" followed by a text input field. To the right of the input field is a button labeled "Senden". A red tooltip with white text "Dies ist ein erforderliches Feld" is displayed below the input field, indicating that the field is required.

# Autofokus

- autofocus-Attribut
  - der Cursor springt direkt in das Eingabefeld
  - darf nur einmal auf der Seite auftauchen

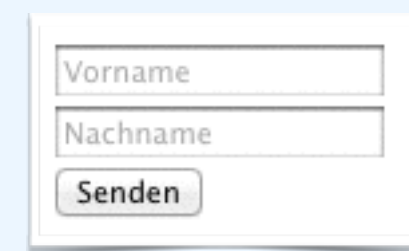
```
<form action="formular-verarbeiten.html">  
  Vorname:  
  <input type="text" name="vorname" />  
  Nachname:  
  <input type="text" name="nachname"  
        autofocus="autofocus" />  
  <input type="submit" />  
</form>
```

Vorname:  Nachname:

# Platzhalter

- In einem Eingabefeld soll eine Beschreibung des Feldes stehen
- Sobald das Feld angewählt wird, soll dieser Text verschwinden
- `placeholder`-Attribut
  - ▶ der Text wird in einer hellen, grauen Schrift dargestellt

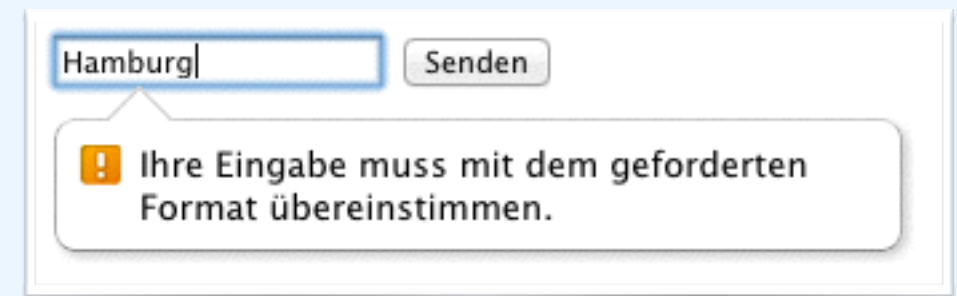
```
<form action="formular-verarbeiten.html">  
  <input type="text" name="vorname"  
        placeholder="Vorname" />  
  <input type="text" name="nachname"  
        placeholder="Nachname" />  
  <input type="submit" />  
</form>
```



# Wertebereich

- Manchmal soll der Wertebereich eines Formularfelds eingeschränkt werden
  - ▶ alle Zeichen sind in Eingabefeldern erlaubt
  - ▶ z.B. haben aber Preise, Postleitzahlen und Email-Adressen ein definiertes Format
- pattern-Attribut erlaubt die Einschränkung mit regulären Ausdrücken

```
<form action="formular-verarbeiten.html">  
  <input type="text" name="plz"  
        pattern="[0-9]{5}"/>  
  <input type="submit" />  
</form>
```



A screenshot of a web form. It features a text input field containing the word "Hamburg" and a "Senden" button. Below the input field, a yellow error message box is displayed, containing an orange exclamation mark icon and the text: "Ihre Eingabe muss mit dem geforderten Format übereinstimmen." This illustrates the result of the regular expression validation defined in the code block to the left.

# Attribute Zusammenfassung

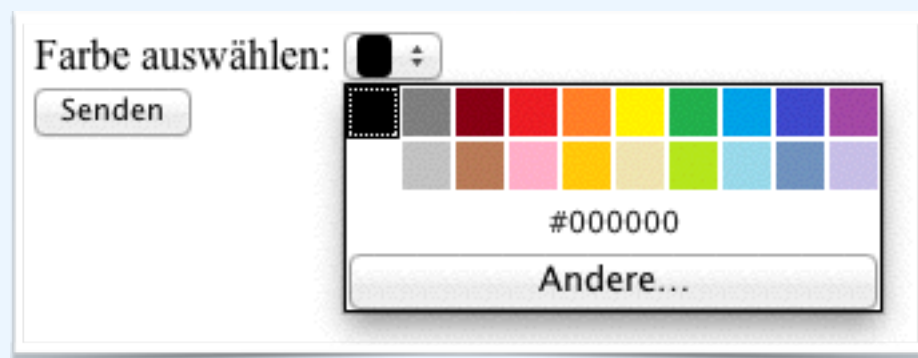
- Die vier Attribute required, autofocus, placeholder und pattern werden schon mäßig gut unterstützt
- Firefox, Chrome und Safari haben vorgelegt
- IE zieht nach
- Mobilbrowser verwehren sich noch

IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
							2.3	
							4	
8	30	35	5.1				4.1	
9	31	36	6.1	12.1	7.1		4.3	
10	32	37	7	24	8		4.4	
11	33	38	8	25	8.1	8	4.4.4	38
	34	39		26			37	
	35	40		27				
	36	41						

# Neue Eingabeelemente

- Die neuen Eingabeelemente sollen (im Sinne der semantischen Elemente) die Bedeutung der Eingabe beschreiben
- z.B. `<input type="color" name="textfarbe"/>`
- Angedacht sind color, date, email, tel, url ...
- Leider sieht die Unterstützung der Browser noch nicht rosig aus

Opera



IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
							2.3	
							4	
8	30	35	5.1				4.1	
9	31	36	6.1	12.1	7.1		4.3	
10	32	37	7	24	8		4.4	
11	33	38	8	25	8.1	8	4.4.4	38
	34	39		26			37	
	35	40		27				
	36	41						

# Zusammenfassung

- HTML5 soll für Formulare viele neue Möglichkeiten bieten
- Vieles ist in den Browsern noch nicht umgesetzt

HTML5 Formulare	HTML4 Formulare + JavaScript
Nur aktuellste Browser unterstützen HTML5	Alle Browser können unterstützt werden
HTML-Formular kann bandbreitenschonend heruntergeladen werden	JavaScript-Bibliotheken müssen heruntergeladen werden
Semantische Informationen sind vorhanden	Semantische Informationen können nicht aus dem HTML-Dokument herausgelesen werden
Browser "garantiert" Validierung der Formulare	JavaScript "garantiert" Validierung der Formulare



# Einbettung von Raster- und Vektorgrafiken

# **Grafikelemente**

# Canvas-Element

- Neues canvas-Element definiert rechteckigen Bereich
- In diesem Bereich kann mit JavaScript "gemalt" werden
  - ▶ 2D pixelgenau (Rasterfläche)
- Falls der Browser das canvas-Element nicht unterstützt, wird der Elementinhalt dargestellt

```
<canvas id="canvas"  
        height="480"  
        width="600">
```

Dein Browser unterstützt kein  
Canvas-Element. Dir entgeht unser  
tolles Anwendungsbeispiel!

```
</canvas>
```

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser
								2.1	
		12.0				3.2		2.2	
	7.0	13.0				4.0-4.1		2.3	
	8.0	14.0	20.0	5.1		4.2-4.3		3.0	
Current	9.0	15.0	21.0	6.0	12.0	5.0-5.1	5.0-7.0	4.0	7.0
Near future	10.0	16.0	22.0		12.1	6.0			10.0
Farther future		17.0	23.0		12.5				

# Canvas-Element

- Außer dem Setzen von Pixeln bietet der Canvas noch folgende Hilfsmittel:
  - ▶ Linien und Bézierkurven
  - ▶ Farbverläufe und Transparenz
  - ▶ Grafik- und Textausgabe
  - ▶ verschieben, rotieren, skalieren von Bereichen
- Canvas schauen wir uns nach der JavaScript-Vorlesung evtl. erneut an
- Beispiel → <http://betermieux.de/beispiele/canvas-tree/>
- weiteres unter → <http://www.canvasdemos.com>



# Canvas Beispiel

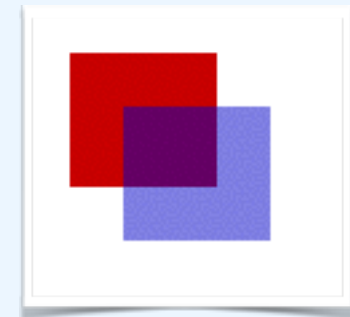
```
<html>
<head>
<script type="application/javascript">

function draw() {
  var canvas = document.getElementById("canvas");
  var ctx = canvas.getContext("2d");

  ctx.fillStyle = "rgb(200,0,0)";
  ctx.fillRect (10, 10, 55, 50);

  ctx.fillStyle = "rgba(0, 0, 200, 0.5)";
  ctx.fillRect (30, 30, 55, 50);
}

</script>
</head>
<body onload="draw()">
  <canvas id="canvas" width="300" height="300"/>
</body>
</html>
```



[https://developer.mozilla.org/en/Canvas\\_tutorial](https://developer.mozilla.org/en/Canvas_tutorial)

# SVG

- Definiert ebenfalls einen rechteckigen Zeichenbereich
  - ▶ 2D vektorbasiert
- Gemalt wird mit Unterelementen aus der SVG-Sprache (XML)

```
<!DOCTYPE html>
<html>
<body>
  <svg width="200" height="200">
    <polygon points="100,10 40,180 190,60 10,60 160,180">
  </svg>
</body>
</html>
```



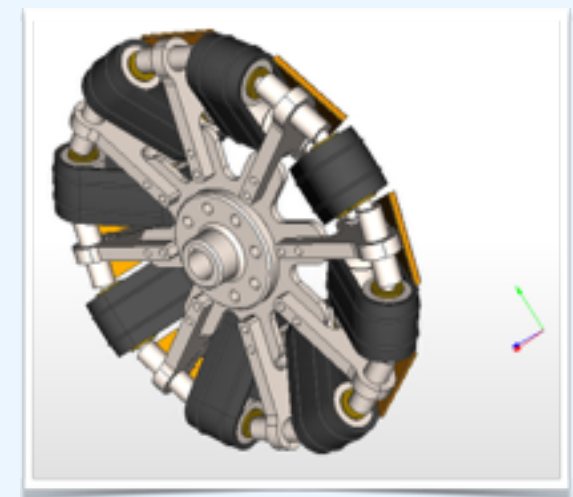
Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser
								2.1	
						3.2		2.2	
		12.0				4.0-4.1		2.3	
	7.0	13.0				4.2-4.3		3.0	
	8.0	14.0	20.0	5.1		5.0-5.1		4.0	
Current	9.0	15.0	21.0	6.0	12.0	6.0	5.0-7.0	4.1	7.0
Near future	10.0	16.0	22.0		12.1				10.0
Farther future		17.0	23.0		12.5				

# Vergleich Canvas - SVG

Canvas	SVG
Auflösungsabhängig (gerastert)	Auflösungsunabhängig (Vektoren)
Wenig Unterstützung für Schriften	Beliebige Schriften
Animation sind schnell	Animationen sind langsam (Änderung am DOM-Baum)
keine Event-Handler	Event-Handler werden unterstützt
keine Barrierefreiheit	Barrierefreiheit möglich
geeignet für Spiele	ungeeignet für Spiele

# WebGL

- Für WebGL wird ebenfalls das canvas-Element verwendet
- In diesem Bereich kann mit JavaScript "gemalt" werden
  - 3D mit OpenGL-Derivat WebGL
- OpenGL ist eine Low-Level-API
  - es gibt Frameworks die Modellierung erleichtern
- Unterstützung ist noch mäßig
  - wenn dann aber Hardware-beschleunigt



Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser
								2.1	
						3.2		2.2	
		12.0				4.0-4.1		2.3	
	7.0	13.0				4.2-4.3		3.0	
	8.0	14.0	20.0	5.1		5.0-5.1		4.0	
Current	9.0	15.0	21.0	6.0	12.0	6.0	5.0-7.0	4.1	7.0
Near future	10.0	16.0	22.0		12.1				10.0
Farther future		17.0	23.0		12.5				

HTML5 Erweiterungen für JavaScript

# **HTML5 - APIs**

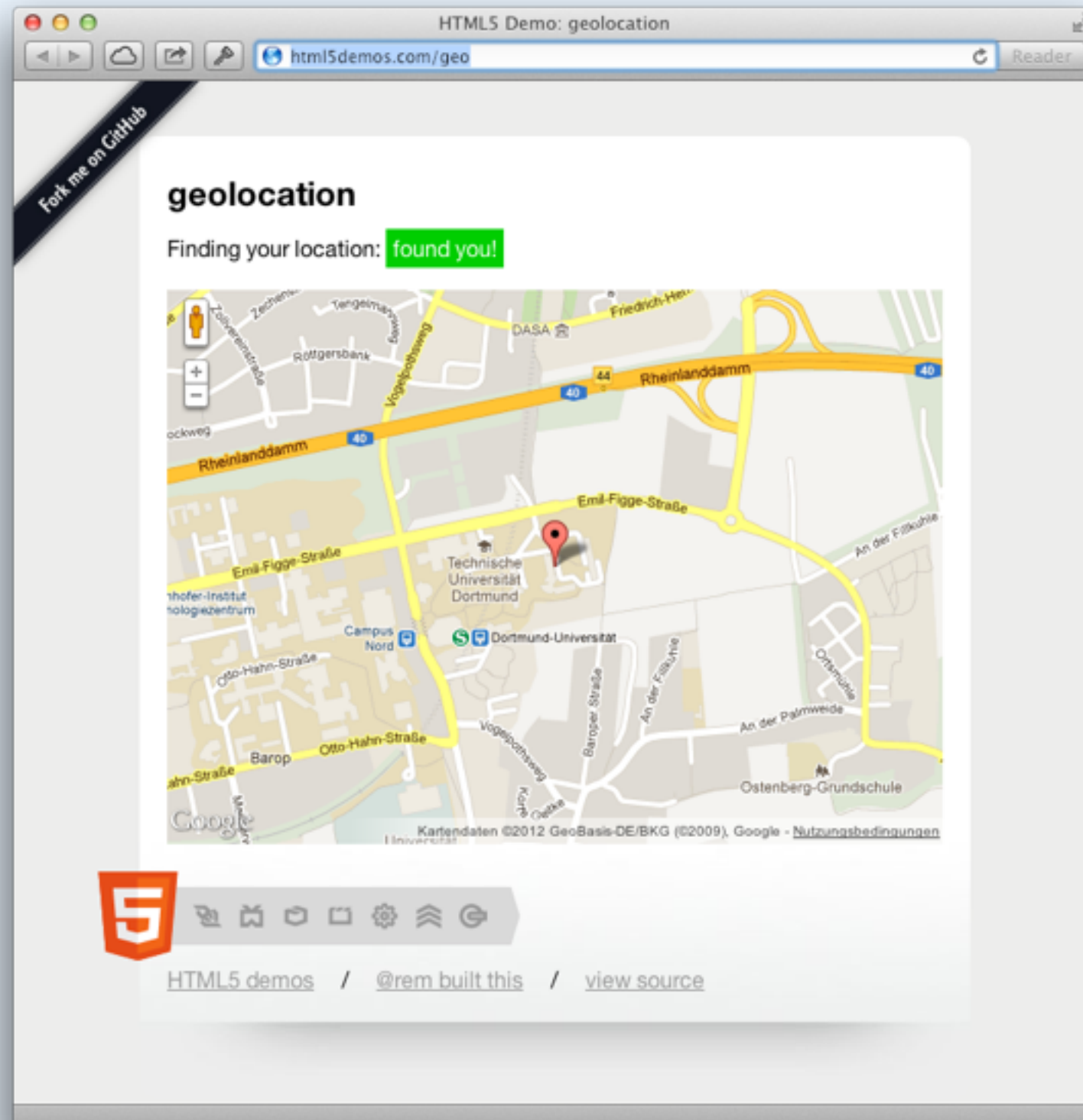


# Geolocation

- Neues `geolocation`-Objekt mit drei Funktionen:
  - ▶ `getCurrentPosition()` → lokalisiere den aktuellen Ort
  - ▶ `watchPosition()` → starte ein dauerhaftes Lokalisieren
  - ▶ `clearWatch()` → beende ein dauerhaftes Lokalisieren
  
- Eine Lokalisierung beinhaltet:
  - ▶ Längen- und Breitengrad
  - ▶ Höhe
  - ▶ Geschwindigkeit
  - ▶ Richtung

<a href="#">Show all versions</a>	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser
								2.1	
		12.0				3.2		2.2	
	7.0	13.0				4.0-4.1		2.3	
	8.0	14.0	20.0	5.1		4.2-4.3		3.0	
Current	9.0	15.0	21.0	6.0	12.0	5.0-5.1	5.0-7.0	4.0	7.0
Near future	10.0	16.0	22.0		12.1	6.0			10.0
Farther future		17.0	23.0		12.5				

# html5demos.com/geo



# Web Storage

- JavaScript-API zur Speicherung von Key-Value Paaren im Browser
- Im Gegensatz zu Cookies können 5MB gespeichert werden (Browserabhängig)
- Same-Origin-Policy (Domains werden isoliert)
- Das globale Objekt `localStorage` bietet die Methoden:
  - ▶ `setItem(key, value);`
  - ▶ `getItem(key);`
  - ▶ `removeItem(key);`

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser
								2.1	
						3.2		2.2	
		12.0				4.0-4.1		2.3	
	7.0	13.0				4.2-4.3		3.0	
	8.0	14.0	20.0	5.1		5.0-5.1		4.0	
Current	9.0	15.0	21.0	6.0	12.0	6.0	5.0-7.0	4.1	7.0
Near future	10.0	16.0	22.0		12.1				10.0
Farther future		17.0	23.0		12.5				

# Sonstiges

- IndexedDB
  - ▶ SQL Datenbank im Browser
- Web Workers
  - ▶ Multithreading in JavaScript
- Web Sockets
  - ▶ bidirektionale Kommunikation mit dem Server außerhalb des Request/Response-Zyklus
- File-API
  - ▶ Zugriff und Bearbeitung von lokalen Dateien



# ZUSAMMENFASSUNG

# HTML5

- Grundlegend anderer Fokus als HTML1 - HTML4
- Wichtige neue Semantikdefinitionen:
  - ▶ Textstruktur
  - ▶ Audio/Video-Elemente
  - ▶ Formularelemente
- Grafikerzeugung im Browser
  - ▶ Canvas, SVG, WebGL
- Programmierschnittstellen
  - ▶ JavaScript-APIs

# Kritik an HTML5

- Viel neues von HTML5 hat mit der Sprache HTML wenig zu tun
  - am Sprachumfang hat sich nur wenig geändert
- HTML5 ist jetzt entkoppelt von einer konkreten Grammatik
  - keine DTD oder Schemas mehr
  - kein Bezug zu XML oder sogar SGML
- An dieser Stelle stand HTML bereits 1993
- Zwanghafte Semantiksuche für alte Elemente (b, i, strong, em)  
"The b element represents a span of text to which attention is being drawn for utilitarian purposes without conveying any extra importance and with no implication of an alternate voice or mood" (WHATWG spec)

# DANKE