Vorlesung

Cloud- und Web-Technologien

(CS3140)

HTML und CSS

Professor Dr. rer. nat. habil. Sven Groppe
https://www.ifis.uni-luebeck.de/index.php?id=groppe

Chronologische Übersicht über die Themen

Nr	Thema	
1	Einleitung	
2	Einführung in das Semantic Web, RDF und SPARQL	Datenmodell
3	Die Semantic Web-Ontologiesprachen RDFS und OWL	
4	Multiplattform-Entwicklung mit Kotlin	Multiplattform
5	Fortgeschrittene Themen mit Kotlin	
6	Einstieg in Cloud Computing, Hadoop	Backend
7	Operatoren der relationalen Algebra in Hadoop	
8	Datenverarbeitung mit Pig	
9	Einführung in Spark und Flink	
10	Stromverarbeitung mit Flink	
11	Knotenzentrische Algorithmen mit Flink	
12	**HTML und CSS**	Web
13	Browserprogrammierung mit JS/JQuery und Serverprogrammierung mit PHP Hypertext Preprocessor	
14	Zusammenfassung und Ausblick	

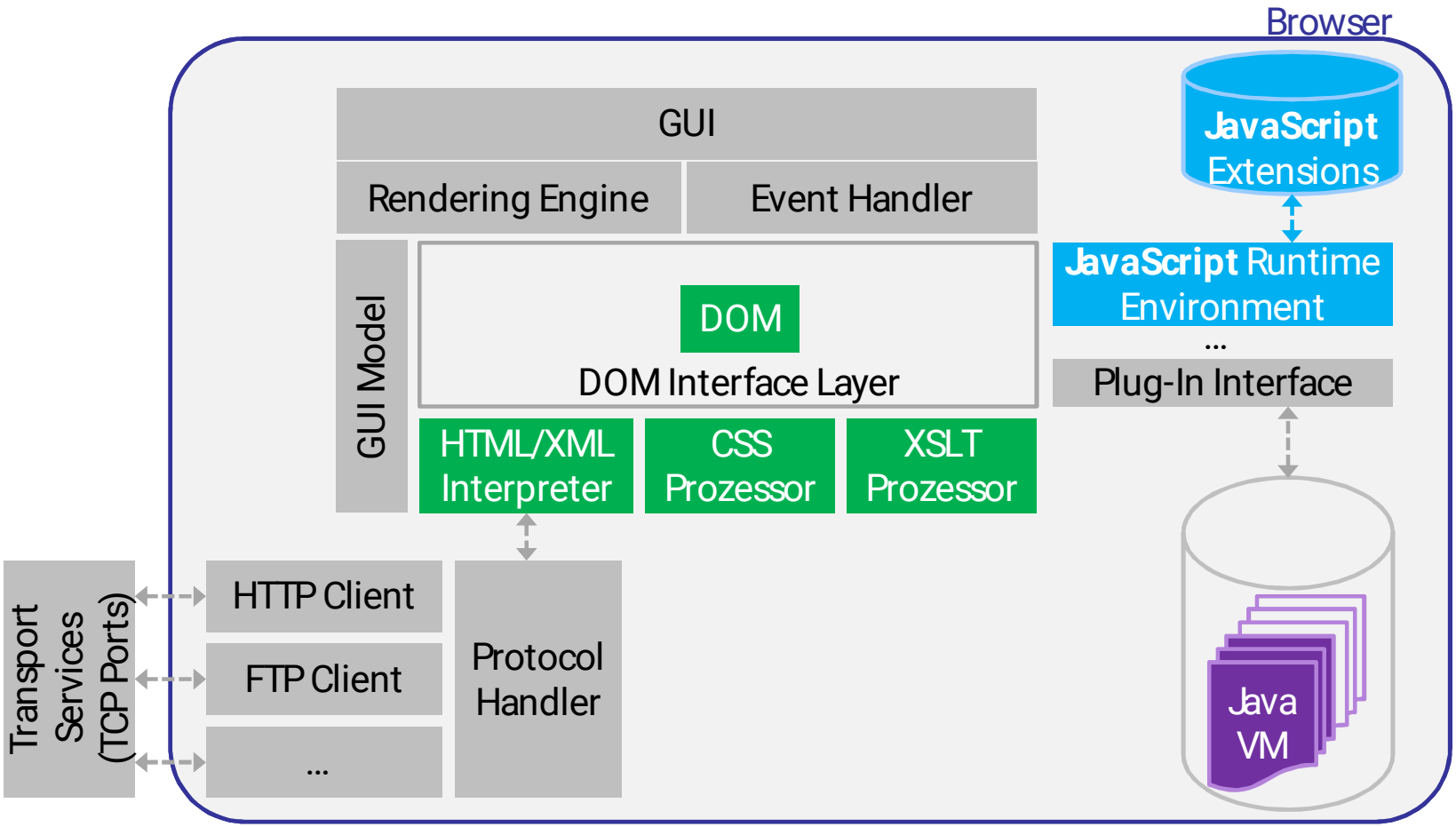
Hypertext Markup Language (HTML)

- textbasierte Auszeichnungssprache zur semantischen Strukturierung digitaler Dokumente
  - Grundlage des World Wide Web für die Darstellung durch Webbrowser
  - Visuelle Darstellung nicht Teil der HTML-Spezifikationen
    - Ausnahme: als veraltet markierte präsentationsbezogene Elemente von HTML
    - ⇝ Gestaltungsvorlagen wie Cascading Style Sheets (CSS)
  - optionale Metadaten: Autor, im Text verwendete Sprachen, ...
- Weiterentwicklung durch
  - World Wide Web Consortium (W3C) und
  - Web Hypertext Application Technology Working Group (WHATWG)

Ziele von Cascading Style Sheets (CSS)

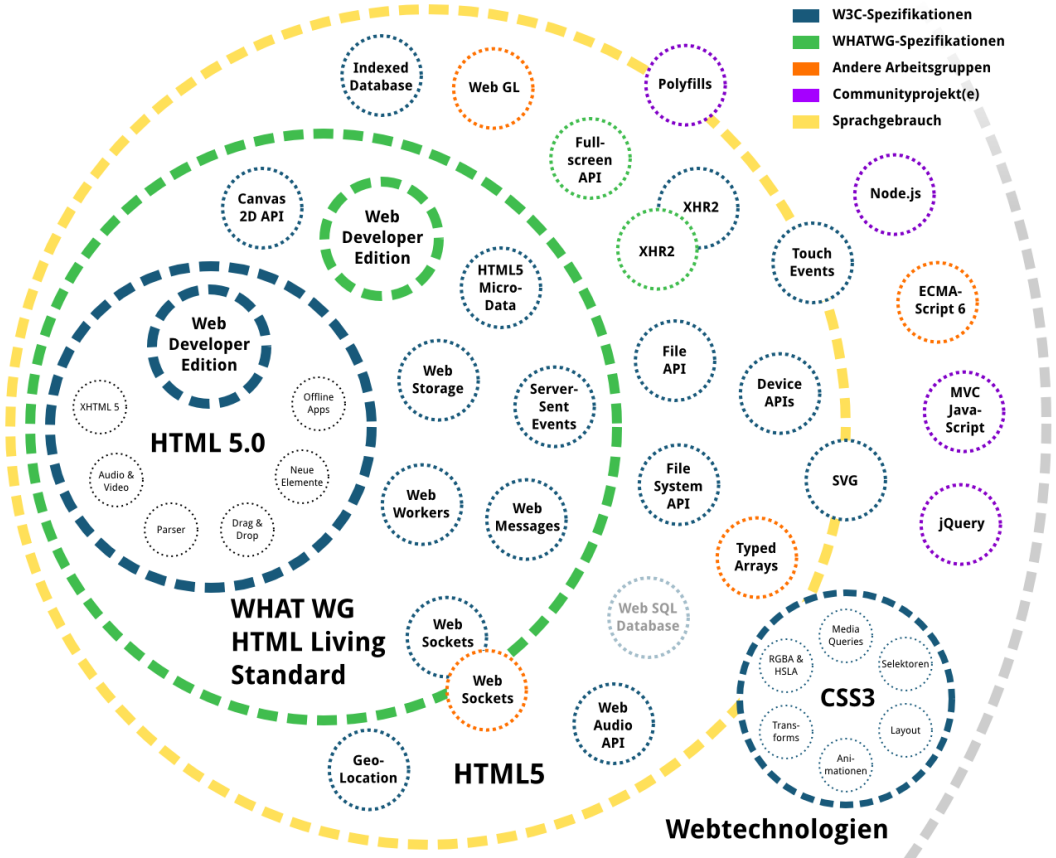
- Trennung zwischen Inhalt und Darstellung von HTML-Dokumenten
- leistungsfähige Layout-Definition für HTML-Dokumente
- Anpassung an verschiedene Ausgabegeräte/-medien
- zentrales Layout-Management

Browser Module



HTML 5 - Überblick über Features

- Web Hypertext Application Technology Working Group (WHATWG)
  - Ziel: Living Standard: eine Spezifikation, die einer ständigen Korrektur und Erweiterung unterliegt (ohne Versionsnummer und -angabe)
  - von mehreren Browserherstellern gegründet
  - Spezifizierte erste Version von HTML 5, die das W3C aufgriff



Grundlegende Syntax 1/2

- Strukturierung von Dokumenten durch Auszeichnungen (engl. *tag*)
  - Öffnende Auszeichnung: `<name>`
  - Schließende Auszeichnung: `</name>`
    - Weglassen in HTML teilweise erlaubt, aber nicht in XHTML (⇝ korrektes Klammergebirge)
  - Öffnende und schließende Auszeichnungen bilden einen Behälter für ihren Inhalt

Grundlegende Syntax 2/2

- `<name></name>` ≡ `<name/>` (Auszeichnung ohne Inhalt)
- Attribute von Auszeichnungen:

  ```
  <name a1="Inhalt1" a2="Inhalt2">...</name>
  ```
- Kommentare: `<!-- … -->`
- Groß-/Kleinschreibung
  bei Element- und Attributnamen:

HTML	XHTML
beliebig	klein

Verhalten des Browsers

- Grundsätze:
  - So viel wie möglich darstellen!
  - Unbekannte Auszeichnungen:
    ⇝ Breche die Gesamtdarstellung nicht ab!
  - Auszeichnungen: nur Empfehlungen zur Darstellung
- Browser ignorieren
  - unbekannte Auszeichnungen
  - Zeilenumbrüche
  - Tabulatoreinrückungen
  - mehrfache Leerzeichen

Grundlegender Aufbau eines HTML-Dokumentes

```
<!DOCTYPE html>
<html lang = "en">
  <head>
    <title> Great Title </title>
    <meta charset = "utf-8" />
  </head>
  <body>
    <p>
      Greetings!
    </p>
  </body>
</html>
```

Dokumenttyp-Deklaration (Angabe der HTML-Version möglich)

Wurzelelement mit Sprachinformation

Kopfelement mit Metainformationen

Titel zur Anzeige in Titelleiste des Browsers

Angabe des verwendeten Zeichensatzes zur Zeichenkodierung

Enthält darzustellenden Inhalt der Webseite (Dokumentinhalt)

Neuer Paragraph

- **XHTML:** Jedes XHTML Dokument muss die nicht grauen Auszeichnungen enthalten
- **HTML:** Auszeichnungen können teilweise fehlen…

Wer findet die 8 Fehler?


```
<!DOCTYPE html5>
<html5 lang = en>
  <head>
   <title> Great Title </t>
   </meta charset = ″utf-8″>
  <head>
  <body>
   <p>
    Greetings!

  </body> <!--  End of Body! ->
 </html5>
```

CSS - „Cascading“

- kombinierte Auswertung unterschiedlicher Arten von Stylesheets
  - Lösung von Konflikten zwischen anwendbaren Layout-Vorgaben mit Rücksicht auf Ursprung, Gewichtung und Spezialisierungsgrad
  - (vom Browser-Hersteller vorgegebenes) Browser-Stylesheet
    - definiert das Standard-Layout eines Browsers für die Elementinstanzen
    - zum W3C-Vorschlag für dieses Stylesheet
  - Benutzer-Stylesheet
    - definiert die Präferenzen eines Benutzers
    - Spezifikation über einen Browser-Dialog
  - Autoren-Stylesheet(s)
    - höchste Priorität bei der Darstellung
    - vom Autor entwickelte(s) Stylesheet(s) zum Layouten seines HTML Dokumentes

Einbindung bzw. Deklaration von CSS

1. In eigener CSS-Datei:

```
<link rel="stylesheet" type="text/css" href="styleheet.css"/>
```

   - <link>-Element darf nur im <head>-Element verwendet werden

2. Im HTML-Dokument:

```
<style type="text/css">
 // … Hier werden die CSS-Regeln definiert …
</style>
```

   - <style>-Element in dieser Form nur im <head>-Element

3. Im style-Attribut einer Elementinstanz:

```
<h3 style="color: red; font: arial">Überschrift!</h3>
```

   - Syntax des style-Attributwertes = Syntax des Deklarationsteils einer CSS-Regel

- Bemerkung: Obige unterschiedliche Arten von Deklarationen sind miteinander kombinierbar!

Geltungsbereich von CSS-Deklarationen

- Global mittels `<link>` oder `<style>` im `<head>`-Element vorgenommene Deklarationen
- Elementinstanz mittels `style`-Attribut
  - Geltungsbereich:
    Elementinstanz einschließlich ihrer Kindselemente
- Geltungsbereich in einem gewissen Scope
  - Ab HTML 5: Stylesheet-Deklarationen im Rumpf mit
    `<style scoped>`
- Konfliktfall:
  lokale Deklarationen haben Vorrang vor globalen
- Stylesheet-Deklarationen innerhalb HTML-Dokument:
  Widerspruch zum Paradigma der Trennung von Inhalt und Darstellung

Stylesheet-Regeln

- Stylesheet als Sammlung von Layout-Regeln
- Aufbau einer Layout-Regel, Bsp.:

```
h1, table { color: red; font: arial }
```

  - Selektoren (hier `h1` und `table`)
    - Auswahl passender Elementinstanzen
    - Angabe mehrerer Selektoren durch Kommata getrennt, obiges Bsp. äquivalent zu:

```
h1    { color: red; font: arial }
table { color: red; font: arial }
```

  - Deklarationsteil (hier `{ color: red; font: arial }`)
    - Menge von (durch Semikola getrennte) Layout-Vorgaben jeweils in Form von Eigenschaftswerte-Paare

Arten von Selektoren

Selektor	Beispiel	Addressierte Elemente
Elementtyp	`h2 {color: red}`	`<h2> … </h2>`
Klassen	`.Classname {color: red}` `h2.Classname {color: blue}`	`<div class="Classname"> … </div>` `<h2 class="Classname Name2"> … </h2>`
ID	`#Identifier {color: red}`	`<h2 id="Identifier"> … </h2>`
Pseudo-Klassen	`a:visited {color: red}`	`<a href="link.html"> … </a>` `link.html` bereits besucht
Pseudo-Element	`p::first-line {color: red}`	`<p> … </p>` 1. Zeile in der Darstellung des Paragraphen

Bemerkungen zu Pseudo-Selektoren

- Pseudoklassen[1]
  - Addressieren von Elementen mit bestimmten Eigenschaften

strukturelle Pseudoklassen (z.B. Pos. im Dokumentenbaum)	dynamische Pseudoklassen (z.B. Benutzeraktionen)	sonstige
:root :empty :first-child :last-child :nth-child() :nth-last-child() :only-child :first-of-type :last-of-type :nth-of-type() :nth-last-of-type() :only-of-type	:any-link :link :visited :hover :active :focus :focus-within :target :disabled :enabled :checked :valid :invalid :in-range :out-of-range	:lang() :not() :matches()

- Pseudoelemente[2]
  - Selektion von **nicht** durch die Markup-Sprache auszeichenbaren Dokumenteninhalt

Pseudoelemente
::first-line ::first-letter ::before ::after ::backdrop ::selection

Attributselektoren

Selektor	Beispiel	Addressierte Elemente
Attributpräsenz	`h2[lang] {color: red}`	`<h2 lang="…"> … </h2>`
Attributwert		
Exakte Übereinstimmung	`h2[lang=de] {color: red}`	`<h2 lang="de"> … </h2>`
In Liste von Werten enthalten	`th[title~=Free]{color: red}`	`<th title="Free Gift">…</th>` `Free` durch Leerzeichen getrennt!
Teilübereinstimmung des Attributwertes		
Attributwert beginnt mit	`td[title^=Fr] {color: red}`	`<td title="Freitag"> … </td>`
Attributwert endet mit	`td[title$=tag] {color: red}`	`<td title="Freitag"> … </td>`
Enthalten in Attributwert	`td[title*=ei] {color: red}`	`<td title="Freitag"> … </td>`

Kombinierte Selektoren

- Selektion von Knotenbeziehungen im Dokumentenbaum

Selektor	Beispiel	Addressierte Elemente
Kind	`h2 > em {color: red}`	`<h2>It's not <em>deep</em></h2>`
Nachfolger	`h2 em {color: red}`	`<h2>It's <span>very <em>deep</em> </span></h2>`
Nächstes-Geschwister	`thead > tr + tr {color: orange}`	`<thead><tr>…</tr> <tr>…</tr>…</thead>`
Nachfolgende Geschwister	`h1 ~ p {color: blue}`	`<p>…</p> <h1>…</h1> <p>…</p> <table>…</table> <p>…</p>`

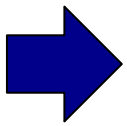
Klassenselektoren

- „Einführung neuer Elementtypen“
  - Meist durch folgende funktionslose HTML-Elementtypen:

Block-Elementtyp	Inline-Elementtyp
…	…

- Beispiel:

```
<html>
<head>
  <title>Example</title>
  <style type="text/css">
     .myheading { font-family: Roboto;
                  color       : blue;
                  font-size   : 20px; }
  </style>
</head>
<body>
  <div class="myheading">Mein Titelstil</div>
  bla…
</body>
</html>
```



Mein Titelstil
bla…

Klassenselektoren - Bemerkungen

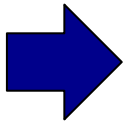
- (traditionelles) HTML: feste Dokumentstruktur
  $\Rightarrow$ Vorgegebene Menge von Elementtypen und deren Darstellung
- 2 Schritte zur Schaffung von Elementinstanzen einer neuen Stylesheet-Klasse
  (mit darüber hinaus ohne weitere intendierte Semantik):
  - Definition einer neuen Klasse mittels `.Classname {...}`
  - Verwendung der neuen Klasse mittels `class="Classname"` in Elementinstanzen des Typs `<div>…</div>` oder `<span>…</span>`
- Warnung des W3C: Intendierte Semantik selbstdefinierter Klassen für Außenstehende oft nicht erkennbar
  ➡ Möglichkeit bitte nicht übertreiben!

CSS-Selektor zum Adressieren von „*folgenden Tabelle*“


```
<!DOCTYPE html>
<html>
  <head>
    <title>WebInfo</title>
    <style type="text/css">

              ??? { font-style: italic }

    </style>
  </head>
  <body>
    <h2>Materialien zu WebInfo</h2>
    <p>Folien entnehmen Sie bitte der
      <span class="it" id="id1">
        folgenden Tabelle
      </span>:
    </p>
  </body>
</html>
```



Materialien zu WebInfo

Folien entnehmen Sie bitte der *folgenden Tabelle* :

Responsive Web Design

- Entwicklung von Webseiten, die die Darstellung an unterschiedlich große Bildschirmgrößen (z.B. von PC, Tablet und Smartphone) anpasst
- Unterstützung durch CSS:
  - Relative Angaben in Prozent, z.B. `width: 33%`
  - „Media Queries“
    - Unterschiedliche Stylesheets für unterschiedliche Bildschirmgrößen:

```
<link href="styles.css" rel="stylesheet"
      media="all and (max-width: 1024px)"/>
```

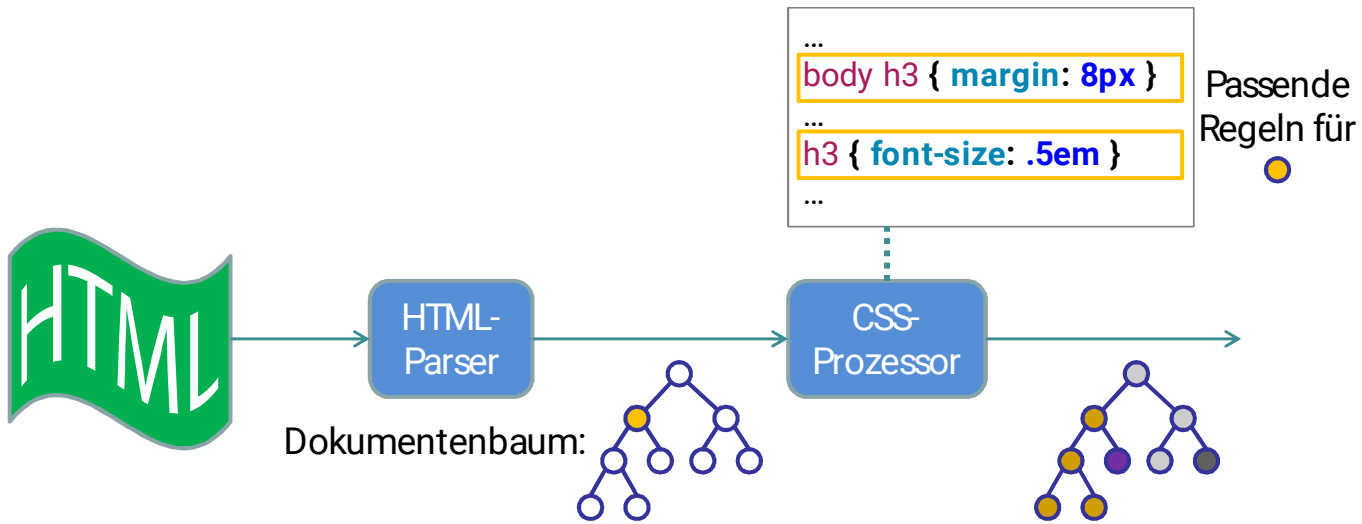
    - Innerhalb eines Stylesheets unterschiedliche Definitionen:

```
@media all and (max-width: 350px) {
      /* hier folgen die Stylesheet-Definitionen für Smartphones */
}
```

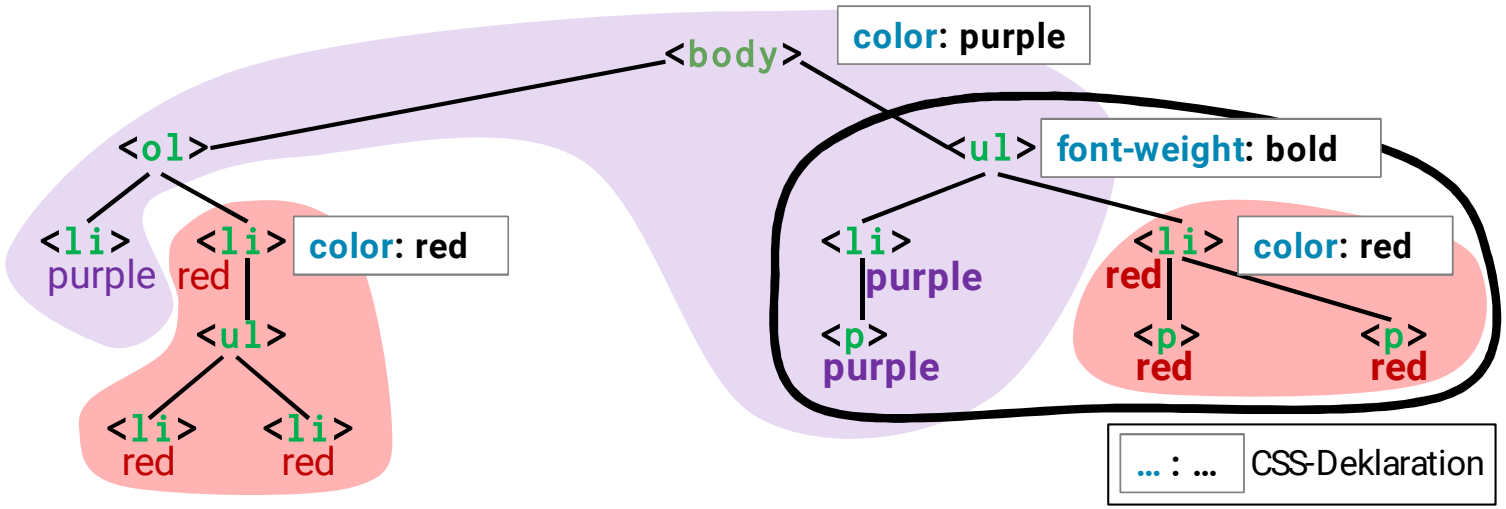
    - Viele weitere Sprachkonstrukte z.B. für Orientierung des Smartphones, Schwarz-Weiss-Displays, 3D-Brillen, …

Darstellung einer Elementinstanz

1. Identifizierung der passenden Regeln in den zugehörigen Stylesheets anhand der Selektoren
2. Anwendung in der Reihenfolge von den weniger zu den mehr spezifischen Regeln
3. Beachtung von Vererbung und Reihenfolge (spätere Regeln vor früheren)



Vererbung von CSS-Deklarationen



- Vererbung von Regeln an eingebettete Elementinstanzen
- Lokale Vorgaben überschreiben vererbte und Default-Werte:

  `color:purple` → `red`

  `font-weight:normal` → **bold**

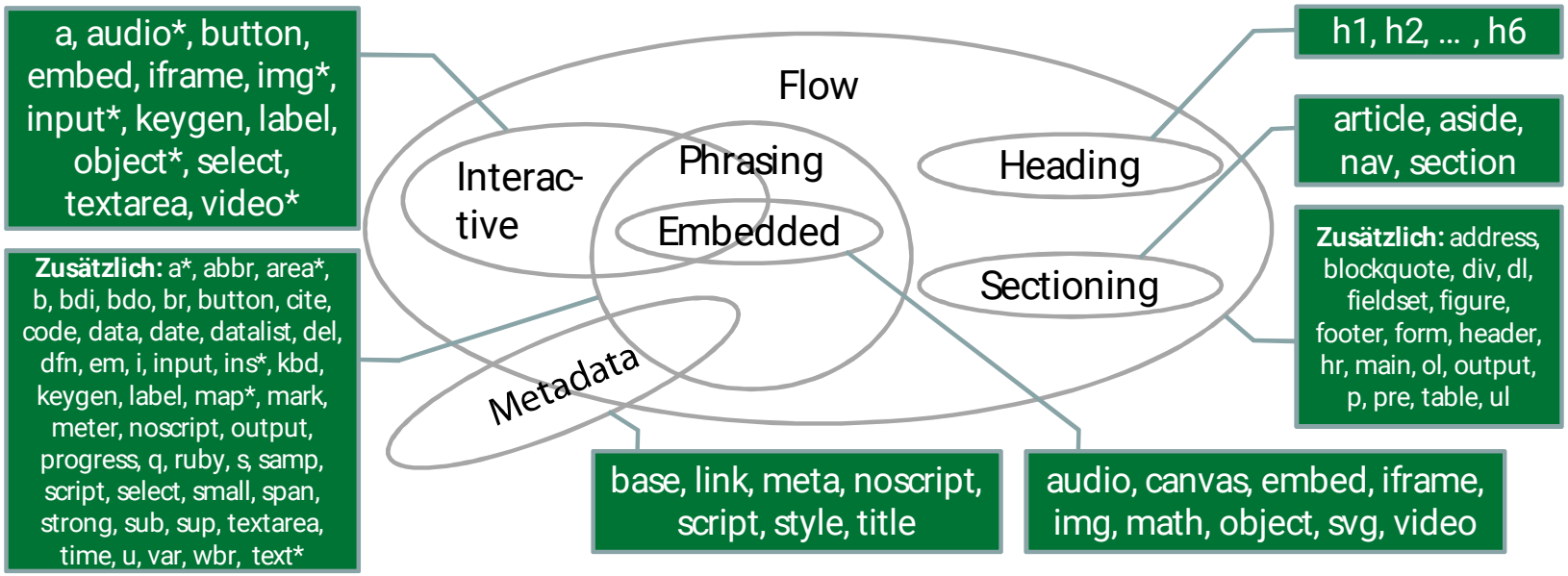
Inhaltsmodelle (bis HTML 4)

Zwei (nicht disjunkte) Klassen von Elementen innerhalb von <body>:

- Block-Elemente
  - erzeugen einen eigenen Absatz im Textfluss
  - beinhalten normalen Text und Instanzen von Inline-Elementen
    - einige auch Instanzen anderer Block-Elemente
  - Beispiele: `<center>` `<div>` `<form>` `<h1>` `<noframes>` `<p>` `<table>` `<ul>`
- Inline-Elemente
  - Platzierung in derselben Zeile wie der vorhergehende Text
  - beinhalten normalen Text und Instanzen weiterer Inline-Elemente
  - Beispiele: `<a>` `<br>` `<cite>` `<em>` `<font>` `<img>` `<small>` `<span>`

Inhaltsmodelle in HTML 5

- Elementinstanzen innerhalb von <body> fallen in mindestens eine der folgenden sieben Klassen:



* Unter bestimmten Umständen

Vergleich der Inhaltsmodelle

HTML 4	HTML 5
syntaktische Aufteilung in Block- und Inline-Elemente	Ablösung und Erweiterung durch eine an semantischen Überlegungen orientierte Aufteilung
Viele (Block-) Elemente dienen der Layout-Gestaltung[1], z.B.: `<center> <frame> <frameset> <noframes>`	Verzicht auf viele dieser (Block-) Elemente[1]
Aus Sicht des Web-Browsers gilt ungefähr die folgende Entsprechung[2]:	
Block-Elemente	~ Flow-Content
Inline-Elemente	~ Phrasing-Content

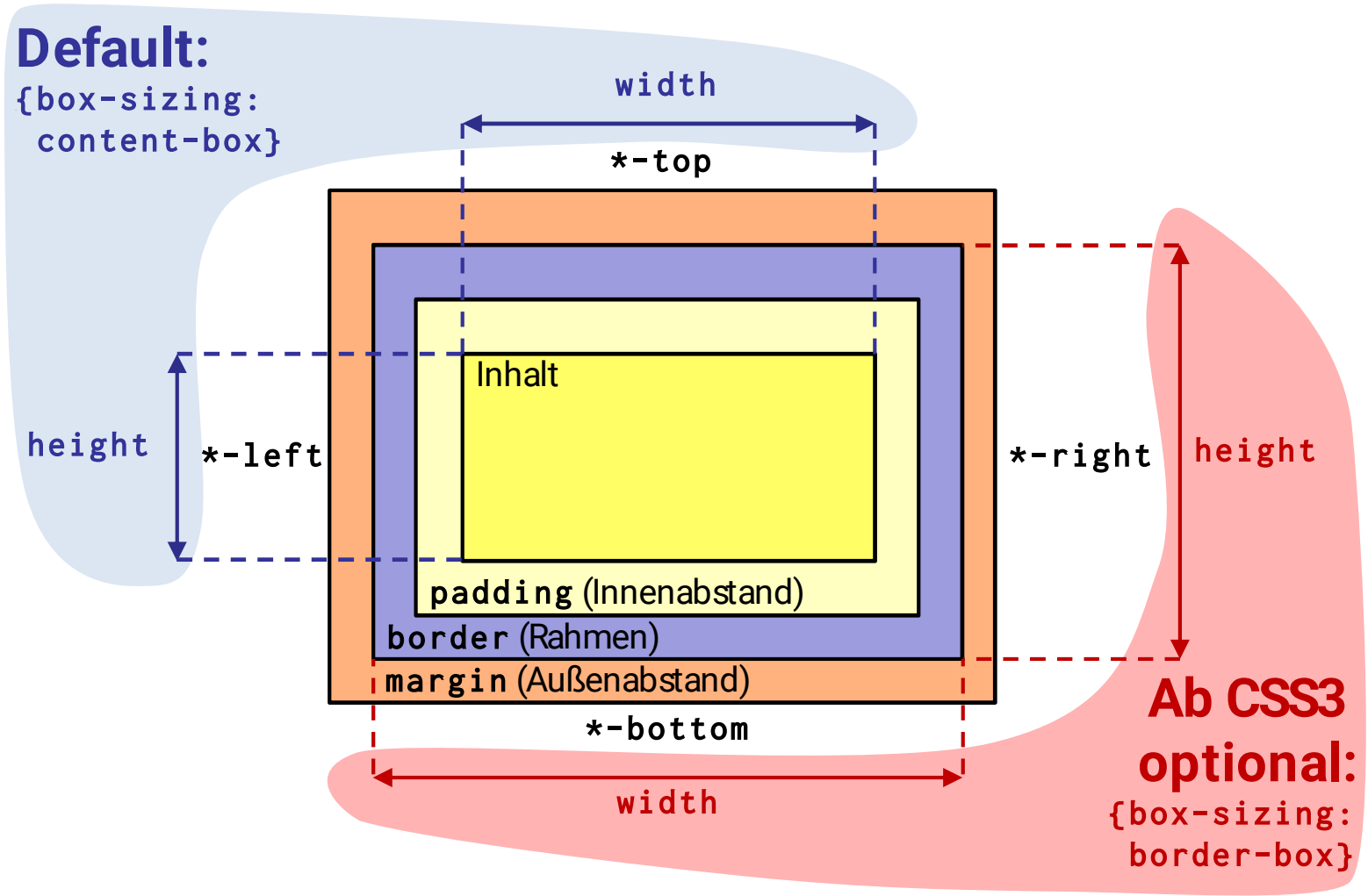
[1] W3 Schools [2] Mozilla Developers

Modell der Visuellen Formatierung

- Jedes Element im Dokumentenbaum generiert 0 oder mehr Boxen
- Jedes dargestellte Element befindet sich in einer Box

  $\Rightarrow$ Rahmen und Außen-/Innenabstände (wenn nicht `0`)
- Verschachtelung von Boxen beliebig möglich
- Vererbung der Layout-Vorgaben von äußeren in innere Boxen

Box-Modell


Default:
{box-sizing:
content-box}
width
*-top
height
*-left
Inhalt
*-right
height
padding (Innenabstand)
border (Rahmen)
margin (Außenabstand)
*-bottom
width
Ab CSS3
optional:
{box-sizing:
border-box}

Arten von Boxen

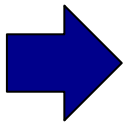
- Steuerung der Art der Box durch `display`-Attribut

Eintrag	Generierte Box
box	Block-Box
`inline-box`	Atomare Inline-Box, die eine Block-Box beinhaltet
`inline`	ein oder mehrere Inline-Boxen (für jede Zeile eine separate)
none	**keine** Box (auch keine unsichtbare), d.h. das Element und dessen Kindselemente (selbst bei anders gesetzten `display`-Attribut) werden nicht dargestellt

- Weitere Block-Arten für Listen und Tabellen

Arten von Boxen - Beispiel

```
<style type="text/css">
  p           { display: inline; }
  span.block  { display: block;
                border: 2px solid #f00;}
  span.inline { display: inline;
                border: 2px solid #000;}
  span.inline_block { display: inline-block;
                      border: 2px solid #000;}
  span.none   { display: none }
</style>
 …
<p>
  This is anonymous text before the block-SPAN.
  <span class="block">Block-SPAN</span>
  This is anonymous text between.
  <span class="inline">This is the content
    <span class="block">of</span>
    inline-SPAN.
  </span>
  This is anonymous text between blocks.
  <span class="inline_block">This is the content
    <span class="block">of</span>
    inline-block-SPAN.
  </span>
  <span class="none">This is not here...</span>
  It follows anonymous text after
  the inline-block-SPAN.
</p>
```



This is anonymous text before the block-SPAN.
Block-SPAN
This is anonymous text between. This is the content of inline-SPAN. This is anonymous text between blocks. This is the content of inline-block-SPAN. It follows anonymous text after the inline-block-SPAN.

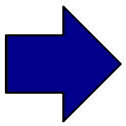
Positionierungsschemata

Fluss	Eigenschaft	Eintrag	Bemerkung
normal	position	`static`	*Default*
		`relative`	Verschiebung* der Box relativ zu seiner normalen Position
absolut		`absolute`	Positionierung* der Box absolut an einer anzugebenden Position
		`fixed`	Box wird zusätzlich nicht mitbewegt beim Scrollen und verbleibt immer an derselben Stelle*
umlaufend	`float`	`left` | `right`	Box wird links bzw. rechts positioniert, Inhalt umläuft diese Box

* Positionierung durch `top`, `bottom`, `left` oder/und `right`-Eigenschaften

Bsp.: absolutes Positionieren und z-Index

```
<style type="text/css">
  .pile {
    position: absolute;
    left: 0.5in;
    top: 0.5in;
    width: 2in;
    height: 2in;
  }
  .greenBackground {
    width: auto;
    height: auto;
    background-color: #0f0
  }
  .z1 { z-index:1 }
  .z2 { z-index:2 }
  .z3 { z-index:3 }
</style> …
<p>
  <img class="pile z1" src="duck.png"/>
  <div id="1" class="pile greenBackground z3">
    This text will overlay the duck image.
  </div>
  <div>This will be beneath everything.</div>
  <div class="pile z2">
    This text will underlay text 1,
    but overlay the duck image. Crazy!
  </div>
</p>
```



Quellen zum Nachschlagen und Nacharbeiten

- HTML
  - MDN, HTML
  - SELFHTML e.V.
  - WHATWG, HTML, Living Standard
  - W3C, HTML 5, Recommendation
  - W3C, HTML Wiki
  - W3 Schools, HTML Reference
- CSS
  - MDN, CSS
  - W3C, CSS Level 2
  - W3C, CSS Specifications
  - W3 Schools, CSS
  - W3C, HTML Tidy
    (standardisieren und säubern von HTML-Code)
  - W3C, Markup Validation Service

Zusammenfassung

- HTML bis Version 4 und HTML 5
  - Grundlegender Aufbau eines HTML Dokumentes
- Ziele von Cascading Style Sheets (CSS)
- Stylesheet-Regeln
  - Geltungsbereich
  - Selektoren
  - Attribute
- Box-Modell und visuelle Formatierung
- Bestimmung anzuwendender Stylesheet-Regeln