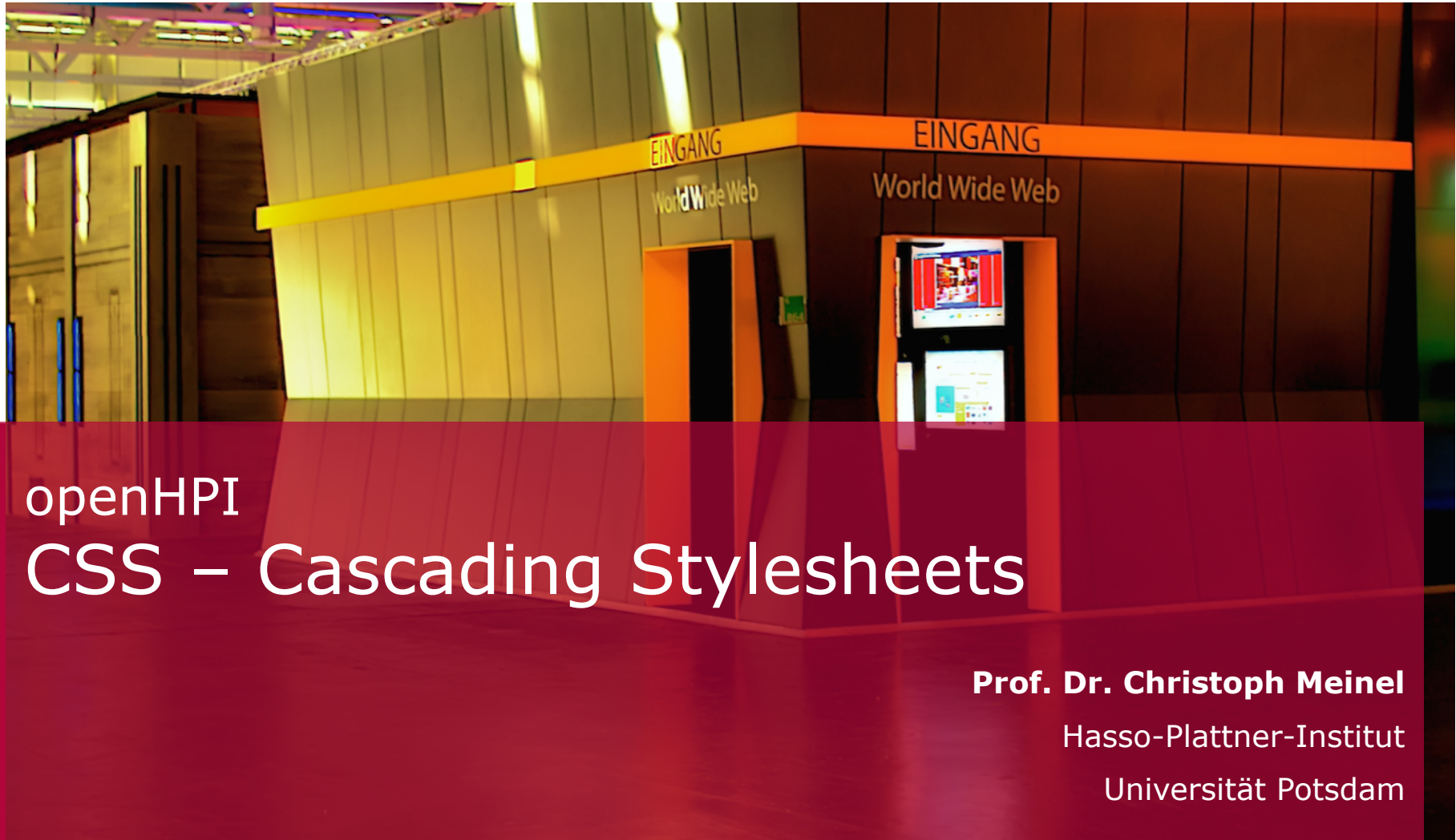




openHPI CSS – Cascading Stylesheets

Prof. Dr. Christoph Meinel

Hasso-Plattner-Institut

Universität Potsdam

Erinnerung (1/2)

- HTML folgt der grundlegenden Idee von SGML:
Trennung von Dokumentstruktur und Dokumentpräsentation
 - SGML beschreibt die Dokumentstruktur
 - Formatierungsbeschreibungssprache DSSSL (Document Style Semantics and Specification Language) definiert das Layout
- Da Browser zu Beginn keine grafische Anzeige besaßen, wurde der Layout-Präsentation bei Webdokumenten kaum Beachtung geschenkt und zu HTML anfänglich keine separate Formatierungs-Beschreibungssprache eingeführt
- Mit dem Aufkommen von grafischen Browseroberflächen – 1993 Mosaic Browser – erfolgten die zunehmend erforderlichen Formatierungsanweisungen direkt in HTML

Erinnerung (2/2)

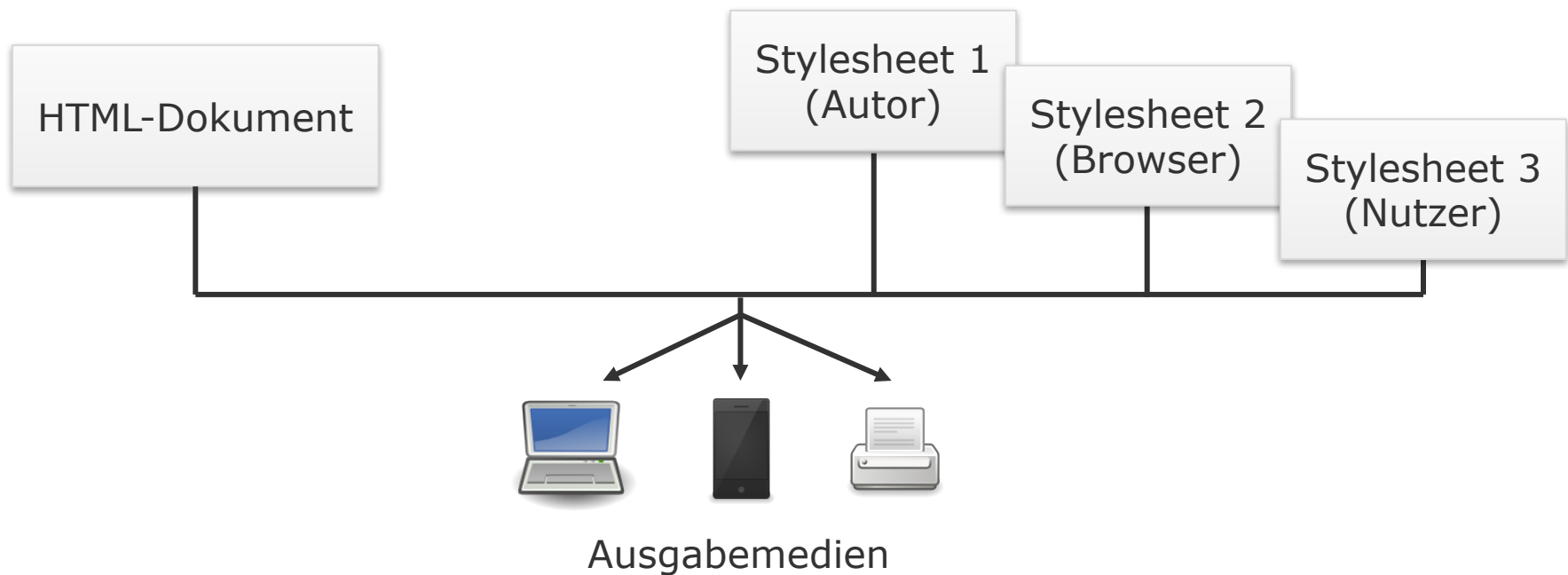
- Rapides Wachstum des WWW führte zum Wunsch, das Dokumenten-Layout so gestalten zu können, das es sicher reproduzierbar und auch an das jeweilige Ausgabemedium angepasst sein sollte
- Mit **Cascading Stylesheets (CSS)** wurde eine unabhängige Formatierungsbeschreibungssprache entwickelt, um **Stylesheets** für die Layout-Präsentation von HTML-Dokumenten beschreiben zu können
- Dabei beschreibt HTML die Dokumentstruktur und Stylesheets legen die Layout-Präsentation der einzelnen Strukturelemente fest für die jeweiligen Ausgabemedien

CSS – Cascading Stylesheets (1/3)

Mit CSS erstellte Stylesheets ermöglichen:

- Einheitliches Layout in zugehörigen HTML-Dokumenten
- Entwurf eines Dokuments nach den Wünschen und Bedürfnissen des Autors oder Nutzers
- Nutzung und Koexistenz von kompletten Stylesheet-Hierarchien, welche vom Autor, Browser-Hersteller oder Nutzer erstellt werden können
- Dokumentenübergreifende Wiederverwendung von Layouts

Verwendung von Stylesheets



CSS – Cascading Stylesheets (3/3)

- CSS bietet direkte Ergänzung zu HTML und weist individuellen HTML-Kommandos Ausgabemedien-spezifische Formatierungen zu
- Haupteigenschaften von CSS sind:
 - exakte Definition des Layouts eines HTML-Dokuments
 - Anpassung an verschiedene Ausgabemedien, z.B.
 - Anweisungen für Druck-Layout,
 - künstliche Sprachausgabe,
 - Smartphones, ...
 - zentrales Layout-Management – Layout-Definition und Aktualisierung kann an zentraler Stelle erfolgen

CSS – Kurze Geschichte

- Erste Spezifikation von CSS (1.0) wurde 1996 vom W3C veröffentlicht zusammen mit Anpassung des HTML 4.0-Standards
- **CSS 2.0** (1998) bietet Ausgabe verschiedener Medientypen – Druck- und Sprachausgabe – und Pixel-genaue Positionierung von integrierten Objekten
- Browser arbeiten (bisher) nicht immer Standard-konform
 - 2009 nahm W3C die Zwischenversion CSS 2.1 als „empfohlenen Kandidaten“ an, um Unregelmäßigkeiten zu korrigieren
- Gleichzeitig begann Entwicklung von CSS 3, eine Version mit modularer Struktur
 - Die Spezifikationsphase für die meisten Teile ist abgeschlossen und Browserhersteller setzten die meisten der empfohlenen Module der W3C um

CSS – Verbindung mit HTML-Dokumenten

Stylesheets können auf verschiedene Arten mit HTML-Dokumenten verbunden werden:

- **Inline-Definition** in HTML-Elementen mit **style**-Attribut, z.B.
 - `<span style="font-weight: bold">Fetter Text</span>`
- **(Interne) Stylesheets** werden in einem HTML-Dokument definiert mit dem **<style>**-Tag (sollte im Header stehen)
 - **Anmerkung:** Hierbei sind Dokumentenstruktur und Präsentation noch nicht strikt getrennt
- **(Externe) Stylesheets** aus separaten Dateien werden eingebunden mithilfe der **<link>**-Deklaration im Header des HTML-Dokuments
- Wenn kombiniert wird, erhalten Inline-Styles Vorrang vor internen Stylesheets (im Header des Dokuments) und externen Stylesheets (lokale Definition vor globaler Definition)

CSS Syntax – Regeln

- **CSS Stylesheet** ist Ansammlung von Regeln die auf ein HTML-Dokument angewandt werden

- Beispiel einer **CSS Regel**:

```
h1 {  
    color: blue;  
    font-family: Arial, Helvetica, sans-serif;  
    font-size: 14px;  
}
```

- **CSS Regel** besteht aus:

- **Selektor** (h1) und
- **Deklaration(en)** ({ color: blue; font-family: [...] })

CSS Syntax – Deklarationen

- **Deklaration** beinhaltet die eigentliche Formatierungsanweisung und weist einer **Eigenschaft** einen oder mehrere **Werte** zu
- Jede CSS-Regel kann **mehrere Deklarationen** enthalten
- Mögliche Formatierungsanweisung beziehen sich z.B. auf die folgenden Elemente:
 - **Schrift** – Schriftart, Schriftgröße, Schriftfarbe
 - **Abstände und Rahmen** – für alle Arten von HTML-Elementen
 - **Position und Größe** – für alle Arten von HTML-Elementen
 - **Farben und Hintergründe** – für alle Arten von HTML-Elementen
 - **Sichtbarkeit und Art der Anzeige** – für alle Arten von HTML-Elementen
 - **Umfließung** – für alle Arten von Block-Elementen
 - ...

CSS Syntax – Selektoren (1/5)

Selektor begrenzt potenziellen Anwendungsbereich von Deklarationen (Formatierungsanweisungen), kurz,

ein Selektor **wählt HTML-Elemente aus**, auf die Deklarationen angewendet werden

Es gibt:

- **Klassen- und ID-Selektoren** – begrenzen den gültigen Bereich auf bestimmtes Element bzw. Gruppe von Elementen gleicher Klasse
- **Attribut-Selektoren** – Elemente mit einem bestimmten Attribut bzw. Attribut-Wert
- **Context-Selektoren** – begrenzen Vererbung von Formatierungsanweisungen
- **Externe Selektoren** (Pseudo-Elemente) – trifft auf Elemente zu, die nicht Teil des HTML-Dokuments sind, aber Teil des Kontexts

CSS Syntax – Selektoren (2/5)

Beispiele für verschiedene Selektor-Typen

- Standard-Selektor **p { color: red; }**
 - Beispiel: `<p>roter Text</p>`

- Attribut-Selektor **p[small] { font-size: 8px; }**
 - Beispiel: `<p small>Das ist eine Fußnote ...</p>`
 - oder: **a[target="_blank"] { color:red; }**
 `<a target="_blank">roter Link</p>`

- Kontext-Selektor **td p { color:#0000FF; }**
 - Beispiel: `<td><p>blauer Text in Tabelle</p></td>`
 - Erläuterung: Absätze (<p>) in Tabellen-Zelle (<td>) werden blau dargestellt

CSS Syntax – Selektoren (3/5)

Beispiele für verschiedene Selektor-Typen

- Klassen-Selektor `p.footnote { font-size: 8px; }`
 - Beispiel: `<p class="footnote">Dies ist eine Notiz</p>`
 - Hinweis: Ein Element kann mehrere Klassen haben, bspw. `<h1 class="blue center">...</h1>`
- ID-Selektor `a#imprint { font-weight: bold; }`
 - Beispiel: `<a id="imprint" href="...">Impressum</a>`
- Hinweis: **class**-Attribut können für **mehrere Elemente** identisch sein, **id** muss in einem Dokument **einzigartig** sein
- Klassen- und ID-Selektoren können ohne Element definiert (und auf verschiedene Elemente angewandt) werden:
 - Beispiele: `.red { color: #FF0000; }`
`#invisible { visibility: hidden; }`

CSS Syntax – Selektoren (4/5)

Gruppieren von Selektoren

- Will man identische Deklarationen für mehrere Selektoren anwenden, kann man diese zusammenfassen
- ```
h1 { font-weight: bold; }
#headline { font-weight: bold; }
```

  
wird zu:  

```
h1, #headline { font-weight: bold; }
```

## Manipulation von Element-Inhalten

- Mit den Selektoren `::before` und `::after` kann vor oder nach dem Inhalt eines Elements Text und/oder CSS ergänzt werden
- ```
p.wichtig::before {
```

`<p class="wichtig">CSS lernen</p>`
 `content: "Wichtig: ";` **Wichtig:** CSS lernen
 `color: red;`
 `font-weight: bold;`
 `}`

CSS Syntax – Selektoren (5/5)

Pseudo-Klassen und Pseudo-Elemente

Einige Selektoren können sich auf bestimmte Zustände von Elementen oder auf bestimmte Untermengen beziehen

- **:hover** – für „Mouseover“-Zustand von Elementen, z.B.
`a:hover { text-decoration: underline; }`
(Links werden unterstrichen, wenn Maus darauf zeigt)
- **:focus** – für aktives Eingabefeld in Formularen, z.B.
`input:focus { border: 1px solid blue; }`
(blauer Rahmen um aktives Eingabefeld)
- **::first-letter** – für den ersten Buchstaben in einem Element, z.B.
`p::first-letter { font-size: 24px; font-weight: bold; }`
(„Initial“: erster Buchstabe eines Absatzes groß und fett)
- Gibt eine Reihe weitere, z.B. `::first-line`, `:first-child`, `:active`, `:required`, `::selection`, ...

Dynamische Selektoren in CSS 3

CSS 3 bietet Reihe neuer **dynamischer Selektoren**, z.B.:

- `E:nth-child(n)` *n*-te Kind-Element vom Element *E*
- `E:nth-last-child(n)` *n*-te Kind-Element gezählt vom letzten
- `E:nth-of-type(n)` *n*-te Kind-Element vom Typ
- `E:nth-last-of-type(n)` *n*-te Kind-Element gezählt vom letzten
- Anstelle von Zahlenwert *n* sind auch Formeln möglich, z.B.:
`li:nth-of-type(2n) { color: red; }`
(Färbt jedes zweite `<li>` rot)

■ Außerdem gibt es in CSS 3 den **Negations-Selektor**

- `:not(E)` alle Elemente außer Element *E*
- `:not(p)` wählt alle Elemente außer `<p>`

CSS Syntax – Vererbung (1/2)

CSS-Formatierungsanweisungen werden im HTML-Dokumentenbaum an alle innerhalb eines Elements liegenden Unterelemente **vererbt**

■ Beispiel:

□ `ol { background-color: red; }`

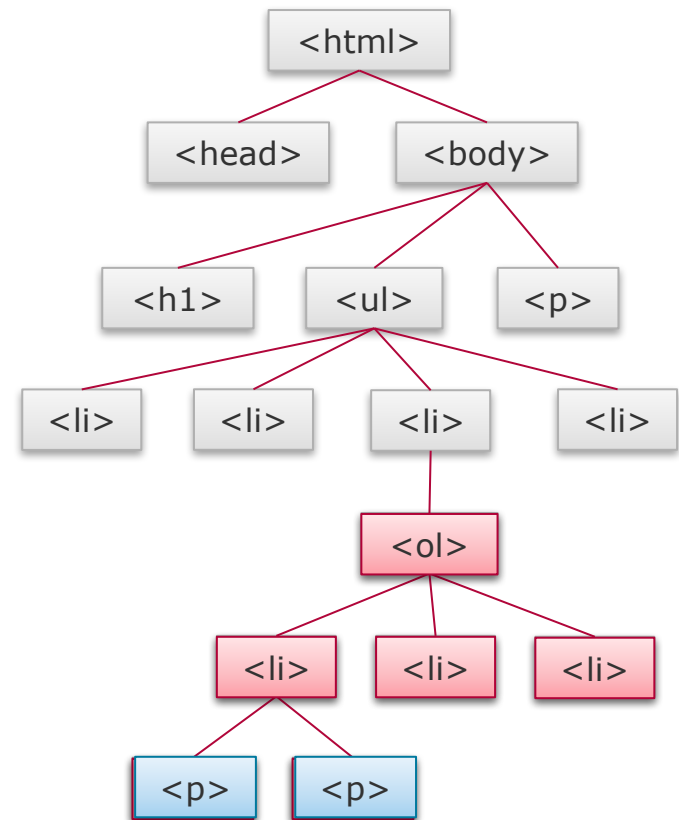
□ Unterbaum von `<ol>` erbt rote Hintergrundfarbe

■ Spezielle Selektoren können Elemente **im Unterbaum** eines Elements auswählen:

□ `ol p { background-color: blue; }`

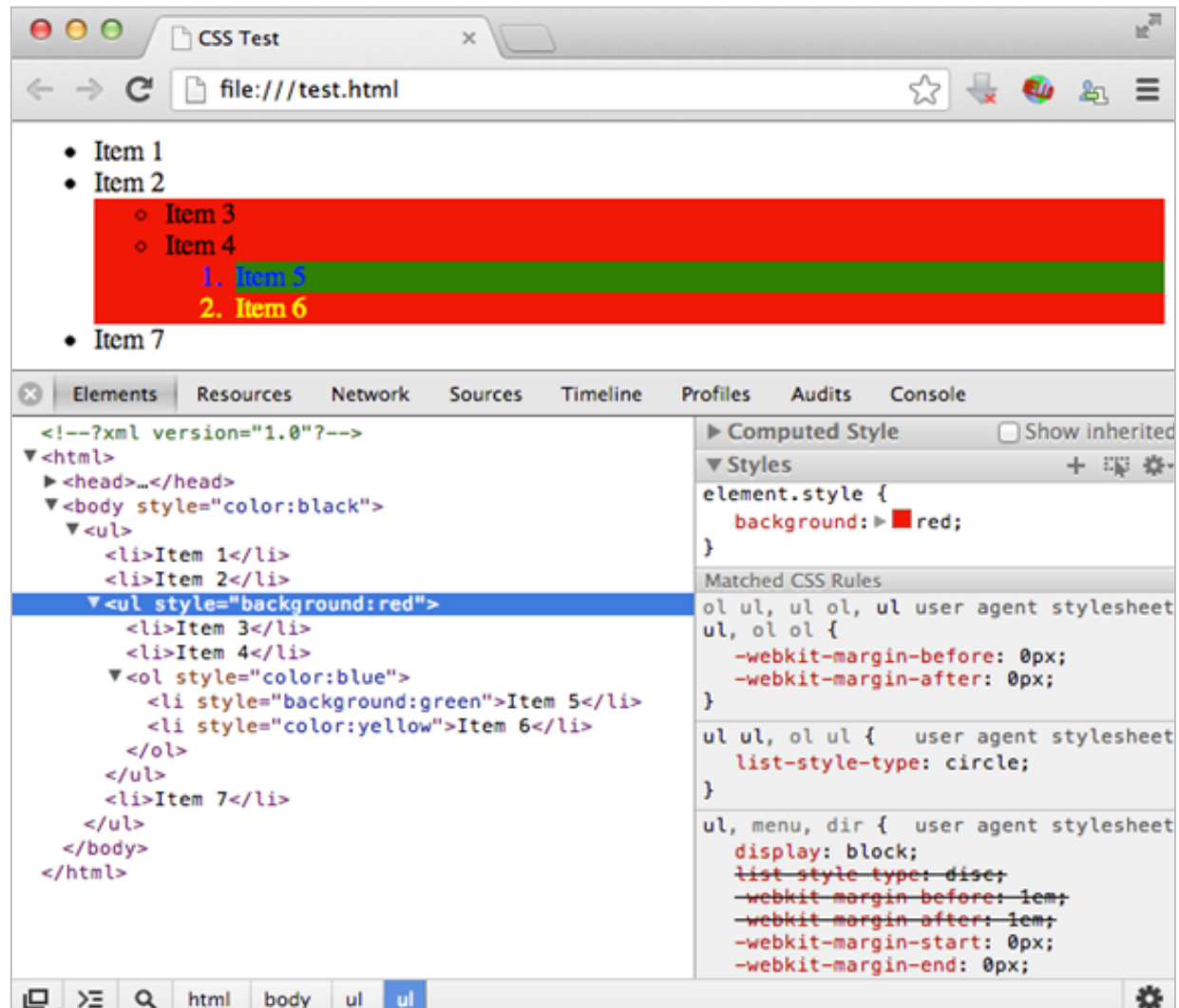
□ Alle `<p>`-Elemente unterhalb von `<ol>` werden blau gefärbt

■ Spezifischere Selektoren überschreiben ererbte Eigenschaften (`ol p` ist spezifischer als `ol` → Hintergrund von `<p>` wird blau)



CSS Syntax – Vererbung (2/2)

Beispiel: CSS Vererbungs- konzept



CSS – Mehrere Stylesheets (1/2)

Eine grundlegende Idee von CSS ist gleichzeitige Nutzung von mehreren Stylesheets. **Mögliche Kaskadierung von Stylesheets** umfasst:

- **Browser-Stylesheets** – in jeder Dokumentenrepräsentation wird ein Browser-spezifisches Stylesheet genutzt
- **Nutzer-Stylesheets** – Browser bieten Nutzern (begrenzte) Möglichkeiten an, die Dokumentenrepräsentation anzupassen (Schrift, Farbe, etc.). Priorität ist höher als Browser-Stylesheets
- **Autoren-Stylesheets** – Autoren haben großen Freiraum in der Layout-Gestaltung; sie können mehrere Stylesheets definieren:
 - modulare Stylesheet Organisation
 - hierarchische Stylesheet Organisation
 - alternative Stylesheets, z.B. für verschiedene Präsentationsformen

CSS – Mehrere Stylesheets (2/2)

Nutzung von mehreren Stylesheets kann zu **Darstellungsfehlern** führen, wenn einem HTML-Element gegensätzliche Formatierungsanweisungen zugewiesen werden – was auch innerhalb eines Stylesheets möglich ist

CSS definiert Reihe von Regeln zur Konfliktlösung:

1. Finde alle relevanten Regeln
2. Ordne Regeln nach ihrer Wichtigkeit – Autoren können Regeln in CSS-Definitionen das Attribut **!important** geben
3. Ordne Regeln nach ihrem Ursprung: Autoren- vor Nutzer- vor Browser-Stylesheets
4. Ordne Regeln nach ihrem Spezialisierungsgrad: Anzahl von ID- und Klassen-Attributen
5. Ordne verbleibende, konkurrierende Regeln chronologisch nach ihrem Auftreten