Tutorial zu Kapitel WT:III

III. Dokumentsprachen

- HTML
- CSS
- XML-Grundlagen
  - XML-Dokumentstruktur
  - Document Type Definition (DTD)
  - Namespaces
- XML-Schema
- XSL-Familie
  - XPath
  - XSLT

Die nach folgenden Erklärungen basieren auf W3C CSS Level 2 Revision 1. Vereinfachungen an verschiedenen Stellen sollen das grundsätzliche Verständnis erleichtern.

CSS

Einbindung von CSS Informationen in HTML Dokumenten

1. Einbindung von CSS-Stylesheets (eigene CSS-Datei) durch Link

   Beispiel:

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

   Das `<link>`-Element darf nur im `<head>`-Element verwendet werden.

2. Einbindung von CSS-Regeln mit `<style>`-Element

   Beispiel:

   ```
   <style type="text/css">
     h3 { color: red; font: arial; }
   </style>
   ```

   Das `<style>`-Element darf in dieser Form nur im `<head>`-Element verwendet werden.

3. Einbindung von CSS-Deklarationen mit `<style>`-Attribut

   Beispiel:

   ```
   <h3 style=" color: red; font: arial">Neues Kapitel</h3>
   ```

   Die Syntax des Attributwertes entspricht dem Deklarationsteil einer CSS-Regel.

CSS

CSS Syntax: Vorbemerkung

0. Beschreibung von zulässigen Zeichenketten in Backus-Naur-Form

   - Regeln beschreiben korrekten Aufbau von Zeichenketten

     zeichenkettenname `::=` *zulässiger-aufbau*

   - Die rechte Seite einer Regel beschreibt den zulässigen Aufbau einer Zeichenkette mit Hilfe von
     - Angabe von Zeichenkettennamen aus anderen Regeln,
     - Angabe einer literal zu verwendenden Zeichenkette in einfachen ' oder doppelten " Anführungszeichen,
     - Angabe eines Zeichen durch eine Auswahlmenge in eckigen Klammern `[ ... ]` (erlaubte Zeichen oder Zeichenbereiche),
     - Reihung (von links nach rechts durch Hintereinanderschreiben),
     - Alternativen (einzelne Alternativen durch `|` getrennt),
     - Anzahl-Constraints (ohne für einmal, `?` für optional, `+` für mindestens einmal, `*` für beliebig oft.

     Alle Beschreibungsmöglichkeiten können kombiniert werden mit Hilfe von Klammerungen mit `(` und `)` für die Eindeutigkeit.

CSS

CSS Syntax: CSS-Stylesheet und CSS-Regel

1. Eine CSS-Stylesheet besteht aus einer Liste von Layout-Regeln.

   Grammatik:

   ```
   stylesheet ::=
           rule*
   ```

2. Eine CSS-Layout-Regel (kurz: CSS-Regel) besteht aus einem Selektorteil und einem Deklarationsteil in geschweiften Klammern.

   Grammatik:

   ```
   rule ::=
           selector_part "{" declaration_part "}"
   ```

   Mit CSS-Layout-Regeln wird das Layout von Elementinstanzen in einem HTML-Dokument beeinflusst.

   Beispiel:

   ```
   h1 { color: red }
   ```

Ergänzungen:

- Stylesheets können auf verschiedene Dateien verteilt werden, die mit Import-Anweisungen unter Angabe der URLs zusammengefügt werden.
  Beispiel:
  ```
  @import "mystyle.css";
  ```
- Stylesheets betreffen unterschiedliche Aspekte des Layout von Inhalten eines Dokumentes. Dazu gehören
  - Links und Lokationen [W3C]
  - Benutzeraktionen [W3C]
  - Zeitverläufe [W3C]
  - linguistische Konzepte [W3C]
  - Eingabezustände [W3C]
  - Dokumentbaumstruktur [W3C]
- Pseudo*elemente* ermöglichen die Selektion von Dokumenteninhalt, der nicht durch die Markup-Sprache ausgezeichnet werden kann. Beispiele: der erste Buchstabe eines Abschnitts, die letzte Zeile eines Abschnitts. [W3C]
- Unter die Rubrik der kombinierten Selektoren fällt insbesondere die Spezifikation von Nachbarschaftsbeziehungen im Dokumentenbaum. Beispiele: ist_Kindknoten_von, ist_Geschwisterknoten_von, ist_Nachfolger_von.

CSS Syntax: Selektoren

3. Der Selektorteil einer CSS-Regel besteht aus einer Komma-Liste von einem oder mehr Selektoren.

   Grammatik:

   ```
   selector_part ::=
           selector ( "," selector )*
   ```

   Der Selektorteil einer CSS-Regel selektiert solche Elementknoten im Dokumentenbaum, die durch (mindestens) einen Selektor aus der Liste der Selektoren selektiert werden.

   Beispiel:

   ```
   h1, h2, h3, h4.myclass { color: red }
   ```

   entspricht den Einzelregeln

   ```
   h1 { color: red }
   h2 { color: red }
   h3 { color: red }
   h4.myclass { color: red }
   ```

CSS Syntax: Selektoren (Fortsetzung)

4. Ein Selektor ist eine Sequenz von einfachen Selektoren oder ein kombinierter Selektor.

   Grammatik:

   ```
   selector ::=
           simple_selector_sequence
         | combined_selector
   ```

   Beachte:

   Kombinierte Selektoren bestehen aus einzelnen Teilselektoren, die mit Kombinatoren (Leerzeichen, $>$, $+$, $\sim$) verbunden sind.

   Eine Sequenz einfacher Selektoren entsteht durch Hintereinanderschreiben einfacher Selektoren ohne Zwischenraum (d.h. ohne Leerzeichen, Zeilenvorschub o.ä.).

   In einer Liste von Selektoren sind die Selektoren durch Komma getrennt.

CSS Syntax: Selektoren (Fortsetzung)

6. Eine Sequenz einfacher Selektoren besteht aus einem Elementtyp-Selektor bzw. dem universellen Selektor gefolgt von beliebig vielen ID-, Klassen-, Attribut-, Pseudo-Klassen-, Pseudo-Element-Selektoren. Der Elementtyp-Selektor bzw. dem universellen Selektor kann fehlen, wenn wenigstens ein weiterer Selektor in der Sequenz enthalten ist.
   Eine Sequenz einfacher Selektoren entsteht durch Hintereinanderschreiben einfacher Selektoren ohne Zwischenraum (Leerzeichen).

   Grammatik:

```
simple_selector_sequence ::=
        type_selector [ id | class | attrib | pseudo | negation ]*
      | [ id | class | attrib | pseudo | negation ]+
```

   Eine Sequenz von einfachen Selektoren selektiert solche Elementknoten im Dokumentenbaum, die durch alle einfachen Selektor in der Sequenz selektiert werden.

   Beispiele:

```
h1[lang=fr].myclass { color: red }
.myclass1.myclass2 { color: green }
```

CSS Syntax: Selektoren (Fortsetzung)

6. Ein Elementtyp-Selektor (oder Typ-Selektor) wird durch den Namen des Elementes oder einen Stern "`*`" angegeben.

   Grammatik:

   ```
   type_selector ::=
           element_name
         | "*"
   ```

   Ein Elementtyp-Selektor selektiert solche Elementknoten im Dokumentenbaum, die den angegebenen Typ (d.h. Namen) haben bzw. beliebige Elementknoten im Fall von "`*`".

7. Ein Klassen-Selektor wird angegeben durch einen Punkt `.` gefolgt von einem Identifier, der als Wert eines `class`-Attributes auftreten darf.
   Grammatik:

   ```
   class ::=
           "." identifier
   ```

   Ein Klassen-Selektor selektiert solche Elementknoten im Dokumentenbaum, die ein `class`-Attribut haben, in dessen zugewiesenen Wert der Name `identifier` auftritt.

CSS Syntax: Selektoren (Fortsetzung)

6. Ein ID-Selektor wird angegeben durch das Hash-Zeichen "#" gefolgt von einem Identifier, der als Wert eines `id`-Attributes auftreten darf.

   Grammatik:

   ```
   id ::=
            "#" identifier
   ```

   Ein ID-Selektor selektiert solche Elementknoten im Dokumentenbaum, die ein `id`-Attribut haben, dessen zugewiesenen Wert der Name `identifier` ist.

7. Ein Attribut-Selektor wird in eckigen Klammern `[...]` angegeben durch einen Identifier, der als Attributname auftreten darf, eventuell gefolgt von Constraints für den Attributwert.

   Grammatik (Ausschnitt):

   ```
   attrib ::=
            "[" attribute_name "]"
          | "[" attribute_name "=" string "]"
   ```

   Ein Attribut-Selektor selektiert solche Elementknoten im Dokumentenbaum, die ein Attribut mit dem Namen `attribute_name` haben. Falls angegeben muss der Wert des Attributs mit `string` übereinstimmen.

CSS Syntax: Selektoren (Fortsetzung)

8. Ein Pseudoklassen-Selektor wird angegeben durch einen Doppelpunkt `":"` gefolgt von einem Identifier, der die Pseudoklasse bezeichnet.

   Grammatik (Ausschnitt):

```
pseudo ::=
          ":link" | ":visited" | ":hover" | ":focus"
        | ":root" | ":first-child" | ":last-child"
        | ":nth-child(...)"
```

   Ein Pseudoklassen-Selektor selektiert Elementknoten im Dokumentenbaum aufgrund dynamischer, struktureller oder anderer Eigenschaften.

   - ❑ `:link / :visited` besuchter / unbesuchter Hyperlink,
   - ❑ `:hover` Element unter Maus,
   - ❑ `:focus` Element mit Focus,
   - ❑ `:root` Wurzelelement,
   - ❑ `:first-child / :last-child` erstes / letztes Kindelement,
   - ❑ `:nth-child(...)` n-tes Kindelement (z.B. `:nth-child(3)` drittes Kindelement).

   Beim Abzählen von Kindknoten zählen NUR ELementknoten.

CSS Syntax: Selektoren (Fortsetzung)

9. Ein Pseudoelement-Selektor wird angegeben durch zwei Doppelpunkte "`::`" gefolgt von einem Identifier, der das Pseudoelement bezeichnet.

   Grammatik (Ausschnitt):

   ```
   pseudo ::=
           "::first-line" | "::first-letter"
   ```

   Ein Pseudoelement-Selektor selektiert Information, die mit anderen Selektoren nicht zugänglich ist und ggf. vom Layout im User-Agent abhängig ist.

   - ❑ `::first-line` erste Zeile des Textes in einem Element im Layout des Browsers,
   - ❑ `::first-letter` erster Buchstabe des Textes in einem Element.

CSS

CSS Syntax: Selektoren (Fortsetzung)

10. Ein kombinierter Selektor ist eine Kombination von Sequenzen einfacher Selektoren mit Hilfe von Kombinatoren "" (Leerzeichen), ">", "+" oder "∼". Eine Kombination von Sequenzen entsteht durch Hintereinandersetzen der Sequenzen mit Kombinator.

    Grammatik:

```
combined_selector ::=
        simple_selector_sequence
                    ( combinator simple_selector_sequence )+

combinator ::= " "   /* descendant combinator */
      | ">"   /* child combinator (new in CSS 2) */
      | "+"   /* adjacent sibling combinator (new in CSS 2) */
      | "∼"   /* general sibling combinator (new in CSS 3) */
```

    Die Auswertung eines kombinierten Selektors erfolgt "von rechts nach links":
    Eine Kombination von Sequenzen selektiert solche Elementknoten im Dokumentenbaum, die durch die letzte Sequenz selektiert werden und in der durch den Kombinator ausgedrückten Relation zu einem Elementknoten stehen, der durch den Anfang des kombinierten Selektors selektiert wird.

    Beispiel: `div em ∼ strong` selektiert `strong`-Elementknoten, die als Geschwisterknoten auf `em`-Elementknoten folgen, die ihrerseits Nachfolger von `div`-Elementknoten sind. (Geschwisterknoten haben denselben Elternknoten.)

CSS

CSS Syntax: Deklaration

11. Im Deklarationsteil werden jeweils durch Semicolon "`;`" getrennt Wertzuweisungen für Properties in Form von "`property : value`" Paaren spezifiziert.

    Grammatik (vereinfacht):

```
declaration_part ::=
        declaration ( ";" declaration )*

combinator ::=
        property ":" value_expression priority?

priority ::=
        "!IMPORTANT"
```

    Die möglichen Properties und jeweils mögliche Wertzuweisungen sind in CSS festgelegt. In CSS 2.1 gibt es 125 Properties. Manche verlangen einen einzelnen Wert, andere erlauben mehrere Angaben.

CSS

Layoutprozess

Für das Layout muss

- für jeden Elementknoten im Dokumentenbaum
- für jede Property
- für das gewünschte Ausgabemedium

ein Wert spezifiziert (specified value) werden.

Ein spezifizierter Wert für eine Property kann

- durch eine CSS-Regel gesetzt werden,
- vom Elternknoten geerbt (inherited) werden oder
- ein festgelegter Anfangswert (initial value) sein.

Genau gesagt, bestimmt man je nach Property und spezifiziertem Wert

- aus dem spezifiziertem Wert ein errechneter Wert (computed value), z.B. durch Umrechnung relativer Maße in absolute Maße,
- im Layoutprozess aus dem errechneten Wert einen benutzten Wert (used value), z.B. die tatsächliche Breite für eine Box, und
- aus dem benutzten Wert den aktuellen Wert (actual value), z.B. durch Approximation von Maßen auf ganzzahlige Pixelwerte.

Die aktuellen Werte werden für die Ausgabe des Dokumentes verwendet.

CSS

Konflikte in CSS

Ein Konflikt besteht in der

- Zuweisung unverträglicher Werte
- an ein und dieselbe Property
- für ein und dieselbe Elementinstanz
- für ein und dasselbe Medium
- in verschiedenen Deklarationen.

Die im Konflikt stehenden CSS-Deklarationen können aus verschiedenen Quellen stammen

- Regeln aus CSS-Stylesheets
- Regeln aus `<style>`-Elementen
- Deklarationen aus `<style>`-Attributen

CSS

Konfliktlösung in CSS

Die Konfliktlösung, d.h. die Entscheidung für eine bestimmte Deklaration geschieht nach drei Kriterien:

1. Quelle und Gewicht (priority)

   `!important`-Deklarationen aus Regeln in Benutzer-Stylesheets
   vor `!important`-Deklarationen aus Regeln in Autoren-Stylesheets
   vor Deklarationen aus Regeln in Autoren-Stylesheets
   vor Deklarationen aus Regeln in Benutzer-Stylesheets
   vor Deklarationen aus Regeln in Browser-Stylesheets

   Deklarationen in `<style>`-Elementen und `<style>`-Attributen werden dem Benutzer-Stylesheets zugerechnet.

2. Spezifität des Selektors: Die Deklaration mit der höchsten Spezifität gewinnt.

   Bei Deklarationen aus Regeln bewertet die Spezifität die "Passgenauigkeit" des Selektors der Regel für die Elementinstanz mit dem Konflikt.
   Deklarationen in `<style>`-Attributen haben den Maximalwert für Spezifität.

3. Reihenfolge der Regeln/Deklarationen: Die letzte Deklaration gewinnt.

   Deklarationen in `<style>`-Elementen und `<style>`-Attributen stehen in der Reihenfolge nach eingebundenen Stylesheets.

CSS

Beispiele für Properties

Property	Values	Initial value	Inherited	Use
`color`	`red, blue, ...`	abh. v. Browser	yes	Textfarbe
`background-color`	`transparent` `red, blue, ...`	`transparent`	no	Hintergrundfarbe der Box

Property	Values	Initial value	Inherited	Use
`font-family`	`sans-serif, ...`	abh. v. Browser	yes	Font-Familie
`font`			yes	Sammelproperty für Font-Eigenschaften
`display`	`block, inline,` `inline-block, ...`	`inline`	no	Layout von Boxen
`position`	`static, relative,` `absolute, fixed`	`static`	no	Position von Boxen
`top, left` `right, bottom`	Wert (mit Maßeinheit / %)	je nach Kontext	no	Offset von Boxen
`margin-top, ...` `..., margin-bottom`	Wert (mit Maßeinheit / %)	0	no	Außenrand von Boxen
`padding-top, ...` `..., padding-bottom`	Wert (mit Maßeinheit / %)	0	no	Innenrand von Boxen
`border-top, ...` `..., border-bottom`			no	Sammelproperty für Rand von Boxen
`width, height`	Wert (mit Maßeinheit / %)	je nach Kontext		Breite von Boxen, Höhe von Boxen

Für alle Properties ist `inherit` als Wert möglich, was für eine Elementinstanz das Erben des Wertes von dem Elternelement erlaubt. Dies betrifft nur die durch den Selektor angesprochenen Elementinstanzen.

Eine komplette Liste aller Properties findet man hier CSS 2.1 Property Index.