Präsentation von XML-Dokumenten mit Hilfe von Stylesheets

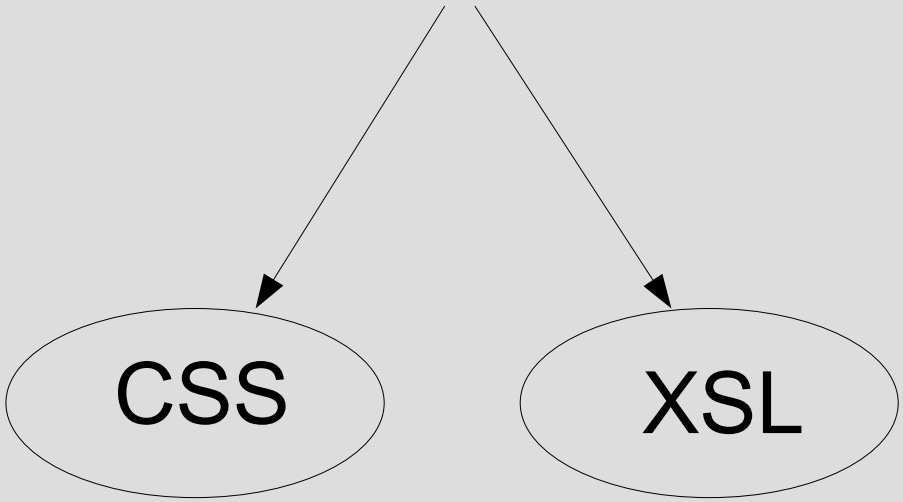
02.05.2006

- Kevin Adrian
- Sandra Aufenberg
- Peter Henschel
- Igor Ionov
- Eva Lai

Stylesheets - Definition

- beschreiben, wie Dokumente angezeigt werden
- trennen die Information von der Darstellung
- interpretieren die zugewiesenen Daten und formatieren sie entsprechend der vorgegebenen Regeln
- sind für XML auf jeden Fall notwendig, da Browser sonst nicht weiß, wie er bestimmte Elemente interpretieren muss

Stylesheets für XML



Geschichte

- 1996: Veröffentlichung von CSS Level 1; erstmals Trennung von Inhalt und Formatierungsanweisungen möglich
- 1996: Verabschiedung von DSSSL (sprich: „Dissel“) als Stilsprache für SGML
- Für XML, das als Teilmenge von SGML entstanden ist, sollte ebenfalls eine eigene Stilkomponente entwickelt werden

Geschichte

- DSSSL zu mächtig, mit CSS allerdings wichtige Funktionen nicht möglich, daher zwei Ansätze zur Entwicklung einer Stylesheet-Sprache für XML
  - Weiterentwicklung von DSSSL-O (Online-Variante von DSSSL) => XS
  - XSL

Geschichte

- 1998: Veröffentlichung von CSS Level 2, allerdings bis heute noch nicht von allen Browsern richtig interpretiert
- 2001: Verabschiedung von XSL als Stilkomponente von XML
- heute: CSS Level 3 ist in der Entwicklung

Cascading Style Sheets

- Nicht XML-basierte Syntax zur Beschreibung von Darstellungseigenschaften in einem Dokument (XML, HTML)
- Inhalt von XML-Dateien bleibt unverändert; es werden keine Transformationen durchgeführt, sondern nur die Stile auf die bestehende Auszeichnung angewandt

Cascading steht für...

- Cascading: (engl. = zusammenfallen) verschiedene Stylesheets für gleiches Element im Dokument möglich
- Unterschiedliche Stylesheets zu einem Element ergänzen sich (Vereiningung)
- Styles werden an Kinder vererbt

Prioritäten

- Wiedersprechen dich die Eigenschaften der verschiedenen Stylesheets, so sind Prioritäten erforderlich, welche Eigenschaften gültig sind
  - 1. Priorität: Style als Attribut imTag notiert
  - 2. Priorität: Style Sheet im Dokument notiert
  - 3. Priorität: Style Sheet in separater CSS-Datei notiert
  - 4. Priorität: Default-Wert, vorgegeben durch den Browser bzw. Parser

Syntax

- CSS besteht aus Darstellungsregeln
- Regel ::= Eigenschaft: Wert
- Regeln werden in Selektoren zusammengefasst
  - Beispiel:

```
Selektor 1, Selektor 2 {
Regel_1;
Regel_2;
Regel_3;
...
Regel_n
}
```

Syntax

- Selektorname = Name des Tags, dessen Darstellungseigenschaften beschrieben werden soll
- Selektor ::= Menge von Darstellungsregeln für ein Element im Dokument
- Selektoren werden mit Komma getrennt, Regeln mit Semikolon
- Style Sheet können sowohl im Dokument als auch in separater CSS-Datei notiert werden

Beispiel

```
Datum {
border: solid black;
background: #dddddd

}
Autor, Titel {
font-family: arial;
font-weight: bold;
color: #00ff00
}
Inhalt {
display: inline;
border-style: hidden;
color: #000088;
margin: 5px;
padding: 5px
}
```

Verbindung mit XML-Dokumenten

- Die Verknüpfung des CSS mit einem XML-Dokument mithilfe der Anweisung:

```
<?xml-stylesheet  Attribute ?>
```

- Es gibt folgende (optionale) Attribute:
  - `type`: MIME Medientyp des Stylesheets, hier bei CSS „text/css“
  - `href`: absolute oder relative Adresse, wo die CSS-Datei zu finden ist
  - `charset`: Gibt das Zeichensatz vom Stylesheet an
  - `title`: Name des Stylesheets
  - `media`: Das vorgesehene Ausgabemedium für dieses Stylesheet

Beispiel

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<?xml-stylesheet type="text/css" href="newsscreen.css"
   title=”Bildschirm” media=”screen” ?>
<?xml-stylesheet type="text/css" href="newsprint.css" title=”Drucker”
   media=”print”?>

<NewsSystem>
<Beitrag>
<Datum>
<Jahr>2006</Jahr>
<Monat>4</Monat>
<Tag>26</Tag>
</Datum>
<Autor>Max Mustermann</Autor>
<Titel>Nachricht1</Titel>
<Inhalt>Eine Nachricht mit Verweis.</Inhalt>
<Verweise>
<HLink href="http://www.nachricht.de">Verweis</HLink>
</Verweise>
</Beitrag>

…

</NewsSystem>
```

Beispiel - Webversion

2006 4 26

Max Mustermann: **Nachricht1**

Eine Nachricht mit Verweis.

Verweis

2005 12 31

Max Mustermann: **Nachricht2**

Noch eine Nachricht

Verweis 1
Verweis 2

2006 1 1

Max Mustermann: **Nachricht3**

Eine Nachricht ohne Verweise.

Beispiel - Printversion

2006 4 26 **Max Mustermann**
Nachricht1

Eine Nachricht mit Verweis.

Verweis <http://www.nachricht.de>

2005 12 31 **Max Mustermann**
Nachricht2

Noch eine Nachricht

Verweis 1 <http://www.nachricht.de>
Verweis 2 <http://www.nachricht.de>

2006 1 1 **Max Mustermann**
Nachricht3

Eine Nachricht ohne Verweise.

Selektoren

Elementselektoren	
*	Jedes Element ausgewählt
A B	Jedes Element B ausgewählt, welches ein Nachfahre vom Element A ist
A > B	Jedes Element B ausgewählt, welches ein direktes Kind vom Element A ist
A + B	Jedes Element B ausgewählt, welches ein Geschwisterelement sofort nach dem Element A ist
Attributenselektoren	
A[B]	Jedes Element A ausgewählt, welches Attribut B besitzt
*[B]	Jedes Element mit Attribut B ausgewählt
A[B="0"] A[B~=0] A[|=B]	Jedes Element A ausgewählt, welches Attribut B mit Wert 0 besitzt. Mit ~= kann man den Wert auch auf ein einzelnes Wort überprüfen, wenn mehrere Wörter durch Leerzeichen getrennt sind. Mit |= schliesslich wird das erste Wort selektiert, wenn mehrere Wörter durch Bindestriche getrennt sind
A#B #B	Jedes Element A ausgewählt, welches einen ID-Attribut mit Wert B besitzt. Steht kein Element vor dem Gatterzeichen, wird die Regel an jedes Element mit ID="B" angewendet

Selektoren

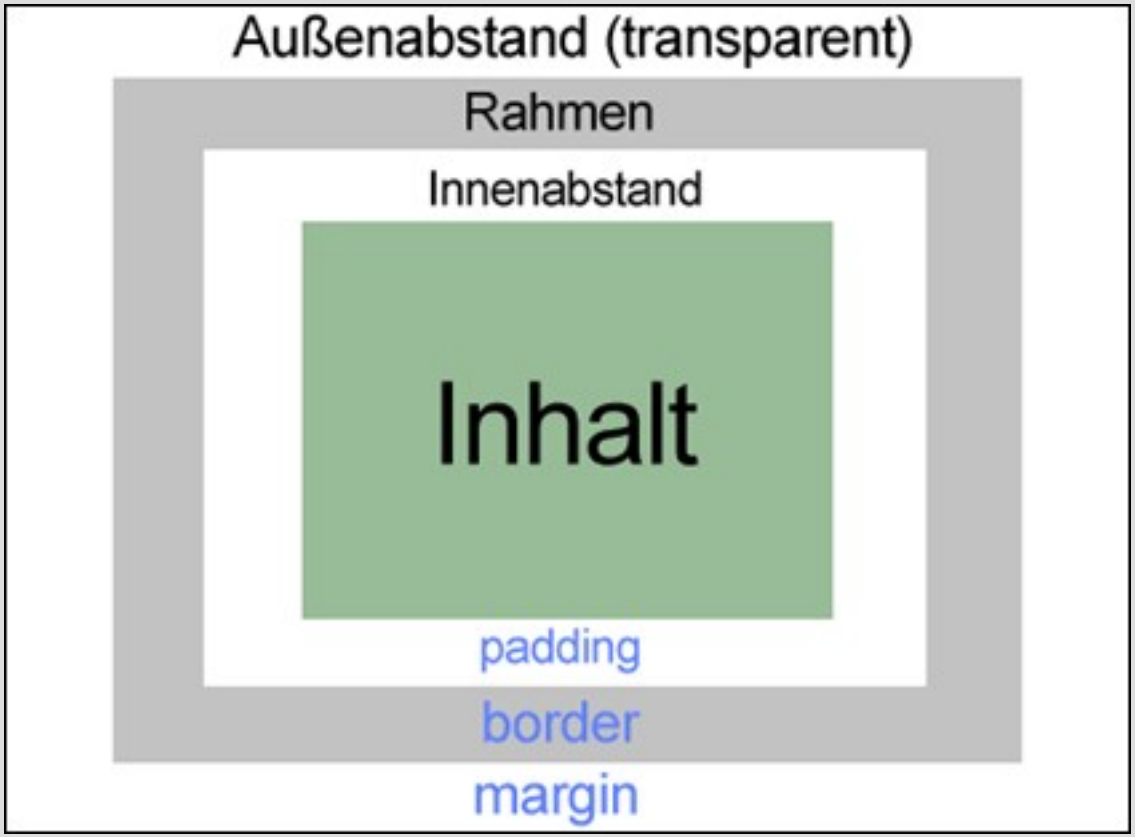
Pseudo-Class Selektoren	
A:first-child	Das erste Kindelement von A wird ausgewählt
A:link	Alle noch nicht besuchten Links im Element A werden ausgewählt
A:visited	Alle bereits besuchten Links im Element A werden ausgewählt
A:active	Das gerade aktives (angeklicktes) Element A wird ausgewählt
A:hover	Das Element A unter dem Cursor wird ausgewählt
A:focus	Das Element A, welches gerade Focus hat wird ausgewählt
A:lang(B)	Das Element A in Sprache B wird ausgewählt
Pseudo-Element Selektoren	
A:first-letter :first-line :before :after	Der erste Buchstabe von A wird ausgewählt Ähnlich - :first-line, :before, :after

Formatierungsmöglichkeiten

- Es gibt grundsätzlich vier Arten von Regeln:
  - Innen- und Außenabstände und Rahmen
  - Farben und Hintergrund
  - Schriften und Texteigenschaften
  - Position und Klassifizierung

Innen- und Außenabstände und Rahmen

- Das Box-Modell



Farben und Hintergrund
Schriften und Texteigenschaften
Position und Klassifizierung

- Zu jedem Element können seine Textfarbe und Hintergrundfarbe definiert werden. Auch Bilder können als Hintergrund eingesetzt werden.
- Es können Schriftenarten, Schriftenstile, Schriftgroesse, Schriftgewicht, Wort- und Zeichenweite, Textdekorationen, Textschatten und Einrückungen sowie Umbruchregeln eingestellt werden.
- Es gibt Regeln, die Position innerhalb des umgebenden Element und die z-position angeben, sowie wie und ob das Element angezeigt wird.

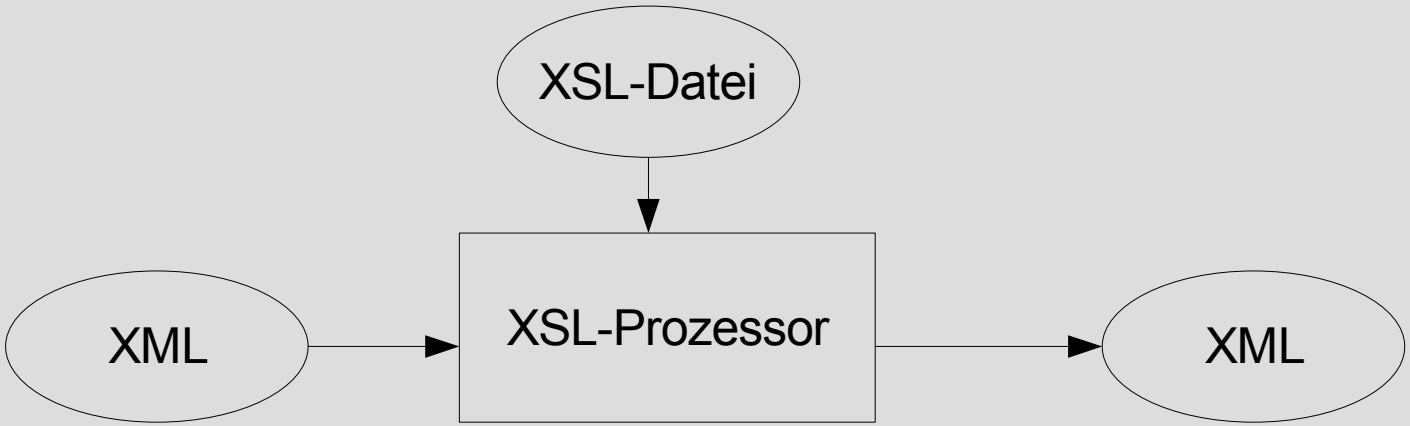
Extensible Stylesheet Language

Extensible Stylesheet Language – XSL

Als W3C-Recommendation am 15. Oktober 2001 erschienen.

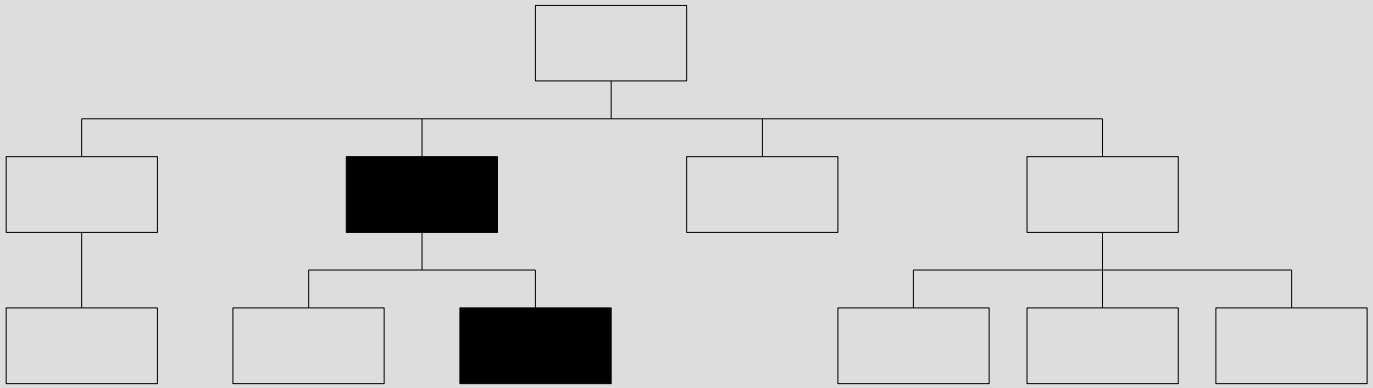
Besteht aus den drei Teilen XSLT, XPath und XSL-FO

XSL Transformations (XSLT)



- Aus einem XML-Dokument wird mittels XSLT ein neues XML-Dokument generiert
- Die XSL-Datei wird in der XML-Datei mit
  `<?xml-stylesheet type="text/xsl" href= beispiel.xsl" ?>`
  aufgerufen

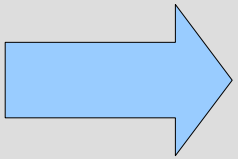
XML Path Language (XPath)



- Finden und Selektieren von Elementen und Attributen (allgemein Knoten) im XML-Baum

XSL Formatting Objects (XSL-FO)

```
<Überschrift>
    ...
</Überschrift>
```



```
<Zentiert>
    <Fett>
        ...
    </Fett>
</Zentiert>
```

- Umformen von XML-Elementen in Formartierungselemente
- Zusätzlicher XSL-FO-Prozessor benötigt

Aufbau von XSLT-Dateien

- Sind XML-Dateien, somit wird eine XML-Deklaration benötigt
  ```
  <?xml version="1.0" encoding="ISO-8859-15"?>
  ```
- Wurzelelement <xsl:stylesheet> gibt an dass es sich um eine XSLT-Datei handelt.
  ```
  <xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  ```
- Enthält mehrere Templates
- Kann weitere Einstellungselemente enthalten (`<xsl:output>`, `<xsl:import>`)

Beispiel einer XSL-Datei

```
<?xml-version="1.0" encoding="ISO-8859-15"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Template für A

Template für B

Template für C

```
</xsl:stylesheet>
```

XSLT-Templates

```
<xsl:template match="xpath">
  <!--
    Neue XML/HTML-Elemente oder weitere
    XSLT-Anweisungen
  -->
</xsl:template>
```

- Der Teil `match="xpath"` wählt die Knoten für das Template aus.
- Der XSL-Prozessor durchläuft alle Templates und wendet diese an.

Beispiel Template

```
<xsl:template match="/">
  <html>
    <head>
      <title>Meine erzeugte Seite</title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>
```

- Weitere Templates werden durch <xsl:apply-templates/> aufgerufen.
- Gewählter Namespace (hier `xsl`) trennt Ausgabeelemente von Steuerelemente.

XSLT-Anweisungen

- `<xsl:apply-templates/>`
  Sucht nach weiteren Templates und wendet diese an.
- `<xsl:if test="expr"> </xsl:if>`
  Prüft einen Ausdruck und wendet dann ggf. den Rumpf an. Negation mit not(expr).
- `<xsl:value-of select="xpath"/>`
  Liest einen Wert von einem Element aus. Mit `"."` wird das aktuelle Element ausgelesen.

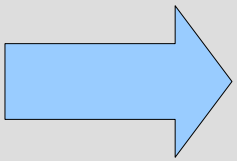
XSLT-Anweisungen

- `<xsl:for-each select="xpath">`
  `</xsl:foreach>`
  Durchläuft eine Menge von Knoten die durch xpath angegeben wurden.
- `<xsl:sort select="xpath"/>`
  Gibt an wie die Knoten der Schleife sortiert werden sollen.

Bäume in XSLT

Quellbaum

```
<NewsSystem>
    <Beitrag>
        Datum>
            <Jahr>2006</Jahr>
            <Monat>4</Monat>
            <Tag>26</Tag>
        </Datum>
        <Autor>...</Autor>
        <Titel>...</Titel>
        <Inhalt>...</Inhalt>
    </Beitrag>
</NewsSystem>
```



Zielbaum

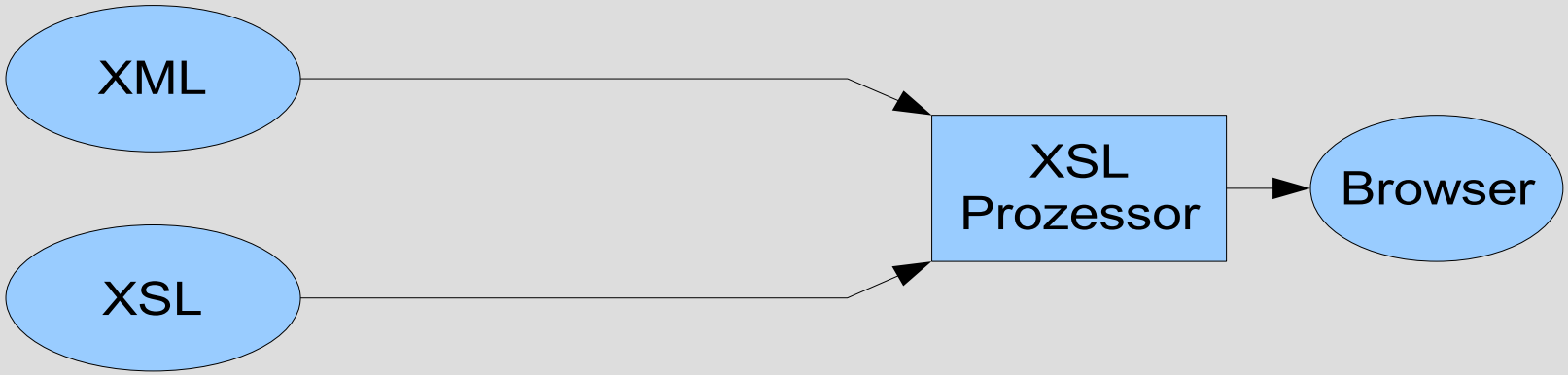
```
<html>
    <head>
        <title>News</title>
    </head>
    <body>
        <div class="news">
            <h1>...</h1>
            <p>
                ...
            </p>
        </div>
    </body>
</html>
```

Beispiel

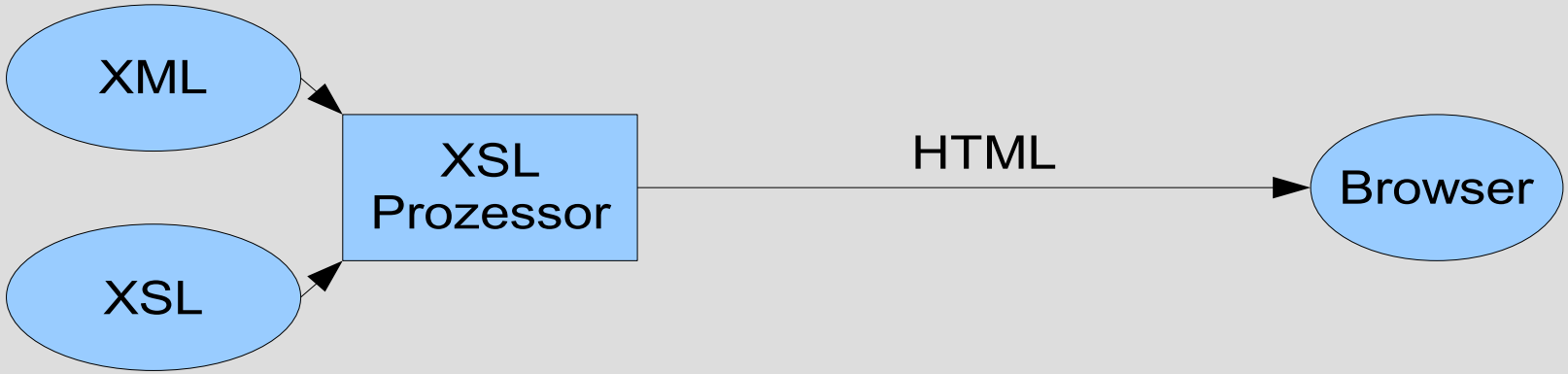
Siehe Browser und Editor

XSLT-Implementierung

Clientseitige Implementierung



Serverseitige Implementierung



XSL vs. CSS

- CSS wird von den meisten Browsern unterstützt, XSL hingegen nicht
- XSL ist mächtiger als CSS
- XSL benötigt einen XML-Parser
- XSL kann das was CSS kann und noch mehr, wie Inhalte neuordnen, verändern und Dinge wie Bilder etc. einfügen
- XSL ist schwerer zu lernen als CSS
- XSL kann man nur auf XML-Dateien anwenden, während CSS sowohl mit XML wie HTML funktioniert
- CSS konzentriert sich auf Webdarstellungen, während XSL häufig auch zum Umwandeln in PDF, RTF, etc. genutzt wird