

# S5 XML Übersicht

**XML** (Extensible Markup Language, dt.: Erweiterbare Auszeichnungssprache)

- seit 1996 vom W3C definiert, in Anlehnung an SGML
- Zweck: Beschreibungen **allgemeiner Strukturen** (nicht nur Web-Dokumente)
- **Meta-Sprache** (“erweiterbar”):  
Die Notation ist festgelegt (Tags und Attribute, wie in HTML),  
Für beliebige Zwecke kann **jeweils eine spezielle syntaktische Struktur** definiert werden (DTD)  
Außerdem gibt es Regeln (XML-Namensräume), um XML-Sprachen in andere **XML-Sprachen zu importieren**
- **XHTML** ist so als XML-Sprache definiert
- Weitere aus XML **abgeleitete Sprachen**: SVG, MathML, XSL-FO, SMIL, WML
- **individuelle XML-Sprachen** werden benutzt, um strukturierte Daten zu speichern, die von **Software-Werkzeugen geschrieben und gelesen** werden
- XML-Darstellung von strukturierten Daten kann mit verschiedenen Techniken **in HTML transformiert** werden, um sie **formatiert anzuzeigen**:  
XML+CSS, XML+XSL, SAX-Parser, DOM-Parser

Dieses Kapitel orientiert sich eng an **SELFHTML** (Stefan Münz), <http://de.selfhtml.org/xml/intro.htm>

# Notation und erste Beispiele

Ein Satz in einer XML-Sprache ist ein Text, der durch Tags strukturiert wird.

Tags werden **immer** in **Paaren von Anfangs- und End-Tag** verwendet:

```
<ort>Paderborn</ort>
```

Anfangs-Tags können Attribut-Wert-Paare enthalten:

```
<telefon typ="dienst">05251606686</telefon>
```

Die **Namen von Tags und Attributen** können für die XML-Sprache **frei gewählt** werden.

Mit Tags gekennzeichnete Texte können geschachtelt werden.

```
<adressBuch>
<adresse>
  <name>
    <nachname>Kastens</nachname>
    <vorname>Uwe</vorname>
  </name>
  <anschrift>
    <strasse>Wiesenweg 37</strasse>
    <ort>Paderborn</ort>
    <plz>33106</plz>
  </anschrift>
</adresse>
</adressBuch>
```

$(a+b)^2$  in MathML:

```
<msup>
  <mfenced>
    <mrow>
      <mi>a</mi>
      <mo>+</mo>
      <mi>b</mi>
    </mrow>
  </mfenced>
  <mn>2</mn>
</msup>
```

# Ein vollständiges Beispiel

Kennzeichnung des Dokumentes als XML-Datei

Datei mit der Definition der Syntaktischen Struktur dieser XML-Sprache (DTD)

Datei mit Angaben zur Transformation in HTML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE produktnews SYSTEM "produktnews.dtd">
<?xml-stylesheet type="text/xsl" href="produktnews.xsl" ?>
<produktnews>
  Die neuesten Produktnachrichten:
  <beschreibung>
    Die Firma <hersteller>Fridolin Soft</hersteller> hat eine neue
    Version des beliebten Ballerspiels
    <produkt>HitYourStick</produkt> herausgebracht. Nach Angaben des
    Herstellers soll die neue Version, die nun auch auf dem
    Betriebssystem <produkt>Ganzfix</produkt> läuft, um die
    <preis>80 Dollar</preis> kosten.
  </beschreibung>
  <beschreibung>
    Von <hersteller>Ripfiles Inc.</hersteller> gibt es ein Patch zu der Sammel-CD
    <produkt>Best of other people's ideas</produkt>. Einige der tollen
    Webseiten-Templates der CD enthielten bekanntlich noch versehentlich nicht
    gelöschte Angaben der Original-Autoren. Das Patch ist für schlappe
    <preis>200 Euro</preis> zu haben.
  </beschreibung>
</produktnews>
```

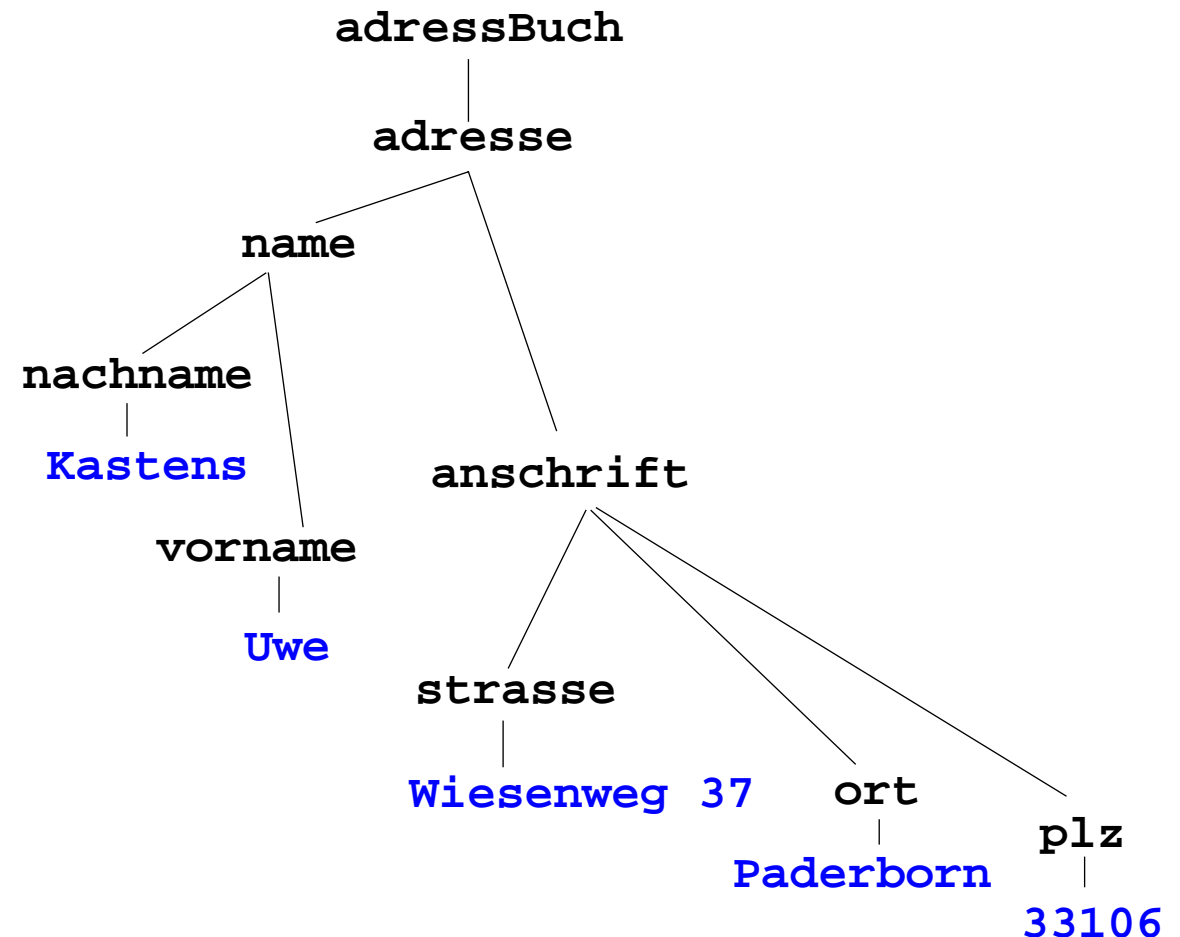
# Baumdarstellung von XML-Texten

Jeder XML-Text ist durch Tag-Paare **vollständig geklammert** (wenn er *wohldefiniert* ist).

Deshalb kann er eindeutig **als Baum dargestellt** werden. (Attribute betrachten wir noch nicht)

Wir markieren die inneren Knoten mit den Tag-Namen; die **Blätter** sind die elementaren Texte:

```
<adressBuch>
<adresse>
  <name>
    <nachname>Kastens
    </nachname>
    <vorname>Uwe
    </vorname>
  </name>
  <anschrift>
    <strasse>Wiesenweg 37
    </strasse>
    <ort>Paderborn</ort>
    <plz>33106</plz>
  </anschrift>
</adresse>
</adressBuch>
```



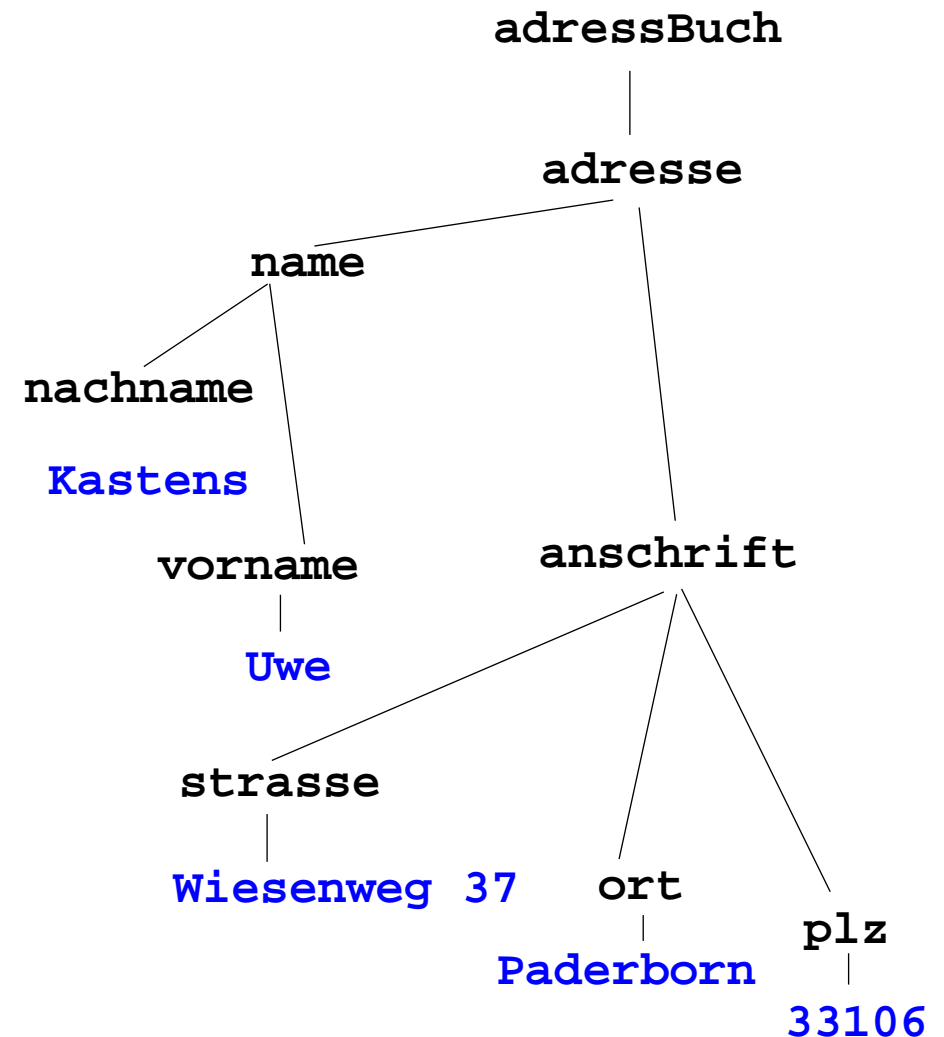
XML-Werkzeuge können die Baumstruktur eines XML-Textes ohne weiteres ermitteln und ggf. anzeigen.

# Grammatik definiert die Struktur der XML-Bäume

Mit **kontextfreien Grammatiken (KFG)** kann man **Bäume** definieren.

Folgende KFG definiert korrekt strukturierte Bäume für das Beispiel Adressbuch:

```
adressBuch ::= adresse*  
adresse   ::= name anschrift  
name      ::= nachname vorname  
Anschrift ::= strasse ort plz  
nachname  ::= PCDATA  
vorname   ::= PCDATA  
strasse   ::= PCDATA  
ort       ::= PCDATA  
plz       ::= PCDATA
```



# Document Type Definition (DTD) statt KFG

Die Struktur von XML-Bäumen und -Texten wird in der **DTD-Notation** definiert. Ihre Konzepte entsprechen denen von KFGn:

KFG	DTD
<code>adressBuch ::= adresse*</code>	<code>&lt;!ELEMENT adressBuch(adresse)* &gt;</code>
<code>adresse ::= name anschrift</code>	<code>&lt;!ELEMENT adresse (name, anschrift) &gt;</code>
<code>name ::= nachname vorname</code>	<code>&lt;!ELEMENT name (nachname, vorname)&gt;</code>
<code>Anschrift ::= strasse ort plz</code>	<code>&lt;!ELEMENT anschrift (strasse, ort, plz)&gt;</code>
<code>nachname ::= PCDATA</code>	<code>&lt;!ELEMENT nachname (#PCDATA) &gt;</code>
<code>vorname ::= PCDATA</code>	<code>&lt;!ELEMENT vorname (#PCDATA) &gt;</code>
<code>strasse ::= PCDATA</code>	<code>&lt;!ELEMENT strasse (#PCDATA) &gt;</code>
<code>ort ::= PCDATA</code>	<code>&lt;!ELEMENT ort (#PCDATA) &gt;</code>
<code>plz ::= PCDATA</code>	<code>&lt;!ELEMENT plz (#PCDATA) &gt;</code>

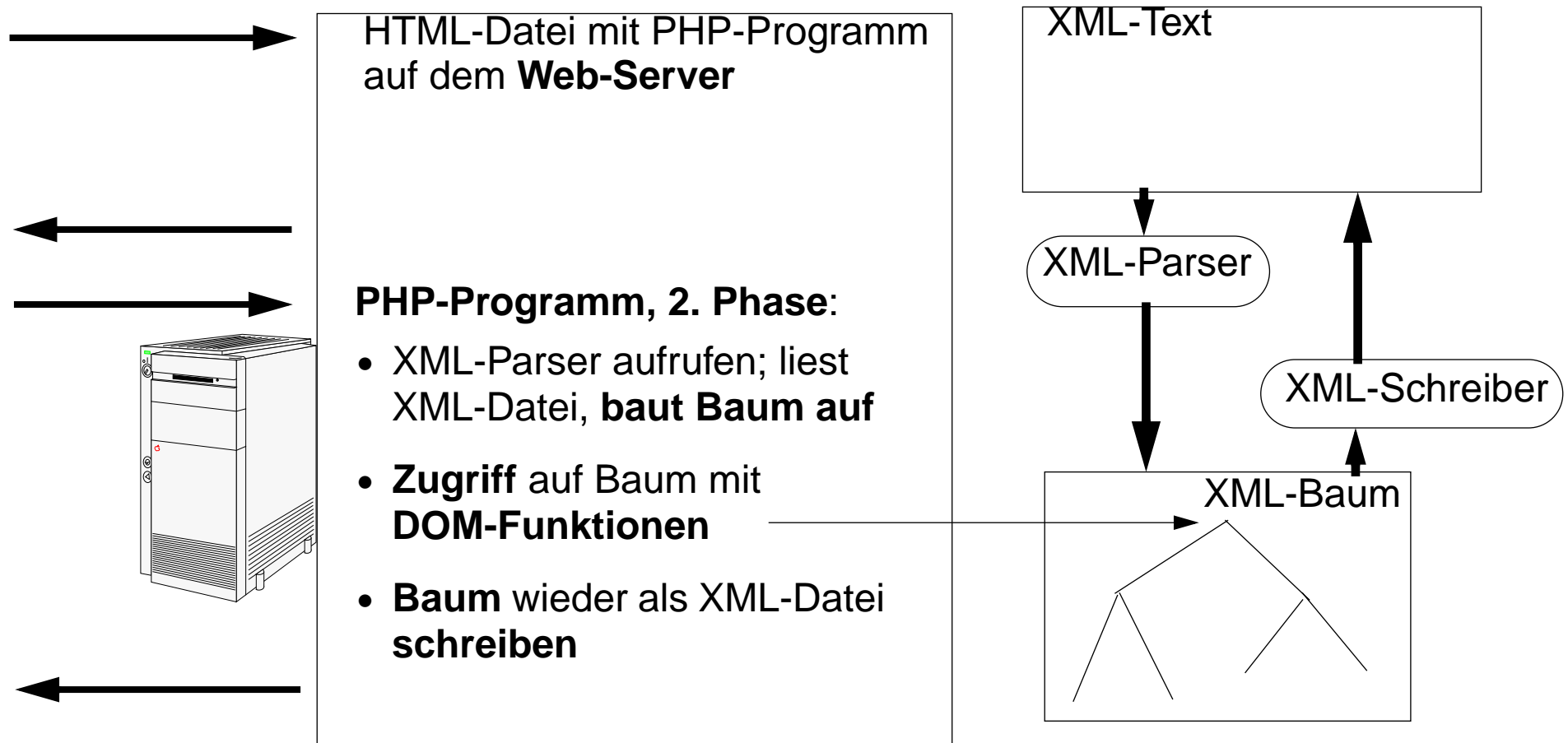
## weitere Formen von DTD-Produktionen:

<code>X (Y)+</code>	nicht-leere Folge
<code>X (A   B)</code>	Alternative
<code>X (A)?</code>	Option
<code>X EMPTY</code>	leeres Element

# XML-Datei als Speicher für strukturierte Daten

Ein **Server-Programm** benutzt eine XML-Datei als Speicher für strukturierte Daten; **liest, ändert und schreibt** sie zurück.

**Anwendungsbeispiel:** Eine Web-Seite sammelt Adressen von potentiellen Kunden. Die 2. Phase des PHP-Programms **trägt Adressen ein** und/oder **zeigt vorhandene an**.



# Präsentation von XML-Daten am Beispiel

**XML-Strukturen** enthalten selbst **keine Information**, die zum **Layout** der Elemente bei einer **Präsentation im Browser** verwendet werden könnte.

Es gibt mehrere Techniken, um **XML-Strukturen zu präsentieren**:

- Der **Browser zeigt den XML-Text mit Tags an** und hebt die Schachtelung hervor - ohne weitere Information möglich
- Den Tags werden mit **CSS Stylesheets Layout-Informationen** zugeordnet.
- Die **XML-Struktur wird in HTML transformiert**, spezifiziert durch **XSL**
- Die **Transformation nach HTML wird in PHP programmiert**, zum Strukturieren des XML-Textes wird ein Parser benutzt.

# Beispiel: Produktinformationen

An diesem Beispiel werden 3 der 4 **Präsentationstechniken** gezeigt. (aus SELFHTML)

## Anwendung:

**Notizen über Produktinformationen** werden in einer Firma als **strukturierte Texte in XML** formuliert, gespeichert, präsentiert und von Werkzeugen verarbeitet (z. B. Mail, News).

## DTD:

```
<!ELEMENT produktnews (#PCDATA | beschreibung)*>
<!ELEMENT beschreibung (#PCDATA | hersteller | produkt | preis)*>
<!ELEMENT hersteller (#PCDATA)>
<!ELEMENT produkt (#PCDATA)>
<!ELEMENT preis (#PCDATA)>
```

## Text-Beispiel:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE produktnews SYSTEM "produktnews.dtd">
<produktnews>Die neuesten Produktnachrichten:
```

```
<beschreibung>
```

```
Die Firma <hersteller>Fridolin Soft</hersteller> hat eine neue Version des beliebten Ballerspiels
<produkt>HitYourStick</produkt> herausgebracht. Nach Angaben des Herstellers soll die neue
Version, die nun auch auf dem Betriebssystem <produkt>Ganzfix</produkt> läuft, um die
<preis>80 Dollar </preis> kosten.
```

```
</beschreibung>
```

```
<beschreibung>
```

```
Von <hersteller>Ripfiles Inc.</hersteller> gibt es ein Patch zu der Sammel-CD <produkt>Best of
other people's ideas</produkt>. Einige der tollen Webseiten-Templates der CD enthielten
bekanntlich noch versehentlich nicht gelöschte Angaben der Original-Autoren. Das Patch ist für
schlappe <preis>200 Euro</preis> zu haben.
```

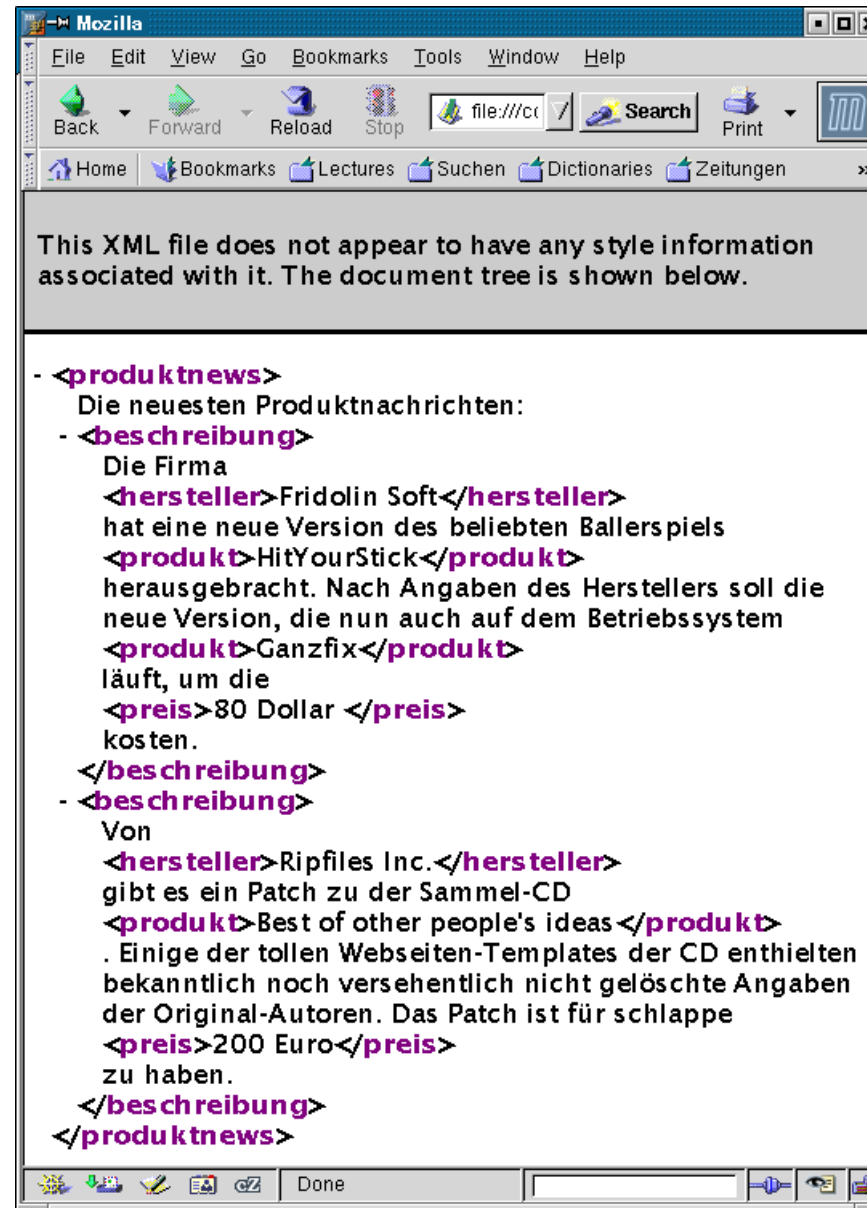
```
</beschreibung>
```

```
</produktnews>
```

# XML-Text direkt präsentiert

## Der Browser

- hebt die Tags hervor,
- zeigt Folgen an,
- zeigt Schachtelung an,
- wendet Strategie für Zeilenumbrüche an.



# XML-Text formatiert mit CSS-Stylesheet

## produktnews

```
{ position:absolute;
  top:10pt;
  left:40pt;
  font-family:sans-serif;
  font-size:18pt;
}
```

## beschreibung

```
{ position:relative;
  display:block;
  width:300px;
  font-size:14pt;
  margin-top:20pt;
  margin-bottom:20pt;
}
```

## hersteller

```
{ font-weight:bold;
  color:blue;
}
```

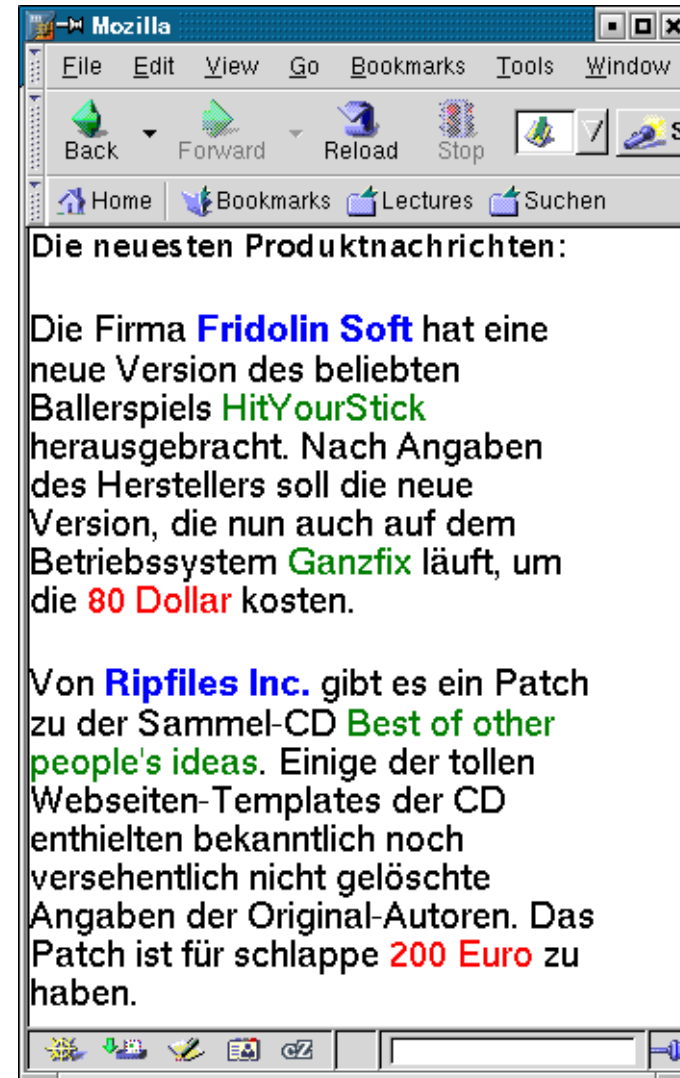
## produkt

```
{ color:green; }
```

## preis

```
{ color:red; }
```

CSS-Stylesheet ordnet den verwendeten **XML-Tags** Formatierangaben zu. Der Browser wendet sie an:



# XML-Text mit XSL in HTML transformiert

```

<xsl:template match="/">
  <html><head>
    <style type="text/css">
      .titel {font-family:sans-serif;
              font-size:18pt; color:green;}
      .absatz {font-family:sans-serif;
              font-size:10pt; color:black;}
      .rot {font-weight:bold; color:red;}
      .blau {font-weight:bold; color:blue;}
    </style></head>
    <body><div class="titel">
      <xsl:apply-templates />
    </div></body></html>
</xsl:template>

<xsl:template match="beschreibung">
  <div class="absatz"><p>
    <xsl:apply-templates />
  </p></div>
</xsl:template>

<xsl:template match="hersteller">
  <span class="rot">
    <xsl:value-of select="." />
  </span>
</xsl:template>
...
<xsl:template match="preis">
  <b><xsl:value-of select="." /></b>
</xsl:template>

</xsl:stylesheet>

```

Ein **XSL-Template** beschreibt die Transformation eines **XML-Elementes** oder der **Baumwurzel** in einen HTML-Text.

Der Inhalt des Elementes wird

- **übersetzt und eingesetzt** oder
- **unverändert eingesetzt**.

