

XML

Eine Einführung

November 2002

Themen

- ◆ **Allgemeine Vorbemerkungen**
 - ◆ **Die Sprache XML:**
 - Grundlagen und Geschichte
 - Sprachgebrauch, Wohlgeformtheit
 - Dokumentstrukturen, DTD, Validierbarkeit
 - ◆ **Erstellung von XML-Dateien:**
 - Editoren: GNU **Exml**, **Emacs**, **XMLSpy**
 - Parser: **SP** von James Clark, **msxml** von Microsoft
 - ◆ **Ausgabemöglichkeiten von XML:**
 - XML-fähige Dateien über WWW-Browser
 - Microsoft **Internet Explorer**, **Mozilla**, **Opera**
 - ◆ **Stilvorlagen und Transformation**
 - **XSL/XSLT**
 - **Cocoon-Projekt** als XML-Framework
-

Dokumentation

- ◆ **XML 1.0 – Grundlagen, Juni 2001, RRZN Skript**
 - ◆ **H. Behme u. S. Mintert: XML in der Praxis, professionelles Web-Publishing mit der Extensible Markup Language**
Addison Wesley, ISBN: 3827313309
 - ◆ **Jon Bosak: XML, Java, and the future of the Web**
sunsite.unc.edu/pub/sun-info/standards/xml/why/xmlapps.html
 - ◆ **Michael Smith: Take My Advice: Don't Learn XML**
xml.oreilly.com/news/dontlearn_0701.html
 - ◆ **Ansonsten: Suchmaschine (www.google.de) ;-)**
-

Verfügbarkeit der Programme

- ◆ **Viele der hier vorgestellten Programme sind frei verfügbar**
 - **Im World Wide Web**
 - **Auf CDROM**
-

Was ist falsch an HTML?

- ◆ **HTML ist daraufhin optimiert, leicht erlernt und angewendet zu werden**
 - **Einheitliches Tag-Set für alle Anwendungen**
 - **Vordefinierte Semantik für jeden Tag**
 - **Verzicht auf die Validierbarkeit zu Gunsten der leichteren Anwendbarkeit**
 - **Ideal für kleinere Web-Anwendungen (*Homepages*) aber viel zu unflexibel für die Organisation größere Datenmengen (*Web-Sites*)**
 - **Wenig Möglichkeit zur modularen Wiederverwertbarkeit**
 - **die Daten lassen sich nur schwer für unterschiedliche Ausgabegeräte präsentieren**
 - **Schwer zu realisierende Navigationsstruktur und der damit verbundenen Pflege**
 - **Bietet keine kontextabhängige Suchmöglichkeit, da keine semantische Auszeichnung**
 - **Datenstrukturen eignen sich kaum für Datenbanken**
-

Was ist falsch an HTML? (cont.)

- ◆ HTML eignet sich gut zur Verteilung von Dokumenten im Web
 - ◆ Die Unzufriedenheit über den Funktionsumfang von HTML demonstriert die stürmische Weiterentwicklung durch Erweiterungen wie
 - Spracherweiterungen: **Frames, Layers** (Netscape), **Marquee** (Microsoft), etc.
 - Scriptsprachen: **JavaScript**, **VB-Script**
 - Programme: **Java-Applets**
 - Animierte Vektorgrafiken: **Macromedia Flash, SVG, VML**
 - Virtuelle Realitäten: **VRML**
 - umfangreiche *Sitemanager* diverser Web-Autorensysteme
 - Dynamische Generierung aus Datenbanken heraus: kommerzielle (Zusatz-)Produkte von DBMS-Anbietern
 - Serverseitige Erweiterung durch **ASP, PHP, ColdFusion, Java-Servlets, JSP, Webobjects**
 - ◆ die fortlaufenden Ergänzungen der Sprache durch die von Herstellern bereitgestellten Tools und Elemente führen leicht ins Web-Chaos
-

Nachteile von HTML

- ◆ **HTML** besteht aus ca. 70 Elementtypen für die Basisstruktur von Texten und Anweisungen für Stilarten
 - ◆ **HTML ist nicht erweiterbar** – die Implementierung eigener Markup-Elemente ist erwünscht, ohne dabei wiederum auf die Implementierung in den Browsern angewiesen zu sein
 - ◆ **HTML ist darstellungsorientiert** – sie bietet zahlreiche Sprachelemente besonders für die Präsentation der Dokumente
 - ◆ **HTML ist weniger wiederverwendbar** und nicht geeignet, für Dokumente, deren Inhalte sich ändern, deren Form aber gleich bleibt; Konvertierungen aus den Textverarbeitungssystemen müssen so immer wieder aufs Neue geschehen
 - ◆ **HTML zeigt die Daten nur in einer Weise** – eine unterschiedliche Sichtweise der Daten in Abhängigkeit der Anforderungen seitens der Benutzer ist kaum möglich
 - ◆ **HTML bietet kaum semantische strukturelle Information** – sie zeigt, wie die Information dargestellt werden soll aber nicht was sie enthält, daraus resultieren auch die Probleme der Suchmaschinen, die richtige Information zu finden
-

Text-Dokumente

- ◆ Seit der Einführung des WYSIWYG und DTP (ca. 1984) orientiert man sich bei der Texterstellung eher am Layout
 - ◆ Dokument besteht aus
 - dem Inhalt = dem eigentlichen Text
 - der visuellen Darstellung = der Formatierung
 - der möglichst maschinenlesbaren logischen Struktur
 - ◆ Ausrichtung nicht auf die Anweisung bzgl. der Formatierung (**procedural markup**), sondern auf die logische Struktur (**generic markup** bzw. **generic coding**)
 - W. Tunncliffe (1967) und Ch. Goldfarb (1969)
 - Goldfarb, Mosher und Lorie entwickelten ende der 60er Jahre bei IBM die Generalized Markup Language (**GML**)
 - SGML als ISO Standard 1986
-

SGML

- ◆ Die verfügbaren Pseudo-Standards: **ASCII**, **RTF** (Microsoft), **PostScript**, Adobe **PDF**
 - ◆ SGML ist ein internationaler Standard (**ISO 8879:1986**) für elektronischen Dokumentenaustausch
 - ◆ Strikte Trennung von Struktur und Formatierung von Dokumenten
 - ◆ **SGML** ist eine Metasprache zur Beschreibung von Dokumentstrukturen und Dokumenttypen
 - ◆ Sprachen, die mit SGML erstellt werden, nennt man Applikationen
HTML ist so eine Applikation von SGML: eine kleine Menge von Elementen, ursprünglich ausgelegt für die Dokumente der Physiker am CERN – später für Web-Dokumente allgemein
 - ◆ Für den Einsatz im World Wide Web ist SGML allerdings viel zu komplex
-

Hypertext

- ◆ Erste Gedanken dazu von Vannevar Bush in seinem Artikel „As we may think“ (1945)
 - ◆ Ted Nelson prägte in seinen Veröffentlichungen zu diesem Thema die Begriffe **Hypertext** und **Hypermedia** (1965)
 - ◆ 1989 legte Tim Berners-Lee den Grundstein zur Entwicklung des World Wide Webs
 - ◆ Zusammenführung der Konzeptionen des Hypertextes mit strukturorientierten Texten auf der Basis von SGML
-

Geschichte von XML

- ◆ Sommer 1996 gründet **Jon Bosak** (*SUN Microsystems*) ein Team zur Entwicklung einer webfähigen SGML-Version
 - Diese XML Arbeitsgruppe stand unter der Schirmherrschaft der W3C (**Dan Connolly**)
 - ◆ November 1996 war das Grundgerüst von XML fertiggestellt
 - ◆ März 1997 entstand Bosaks Aufsatz: „XML, Java and the Future of the Web“
 - ◆ Dezember 1997 wurde das erste Proposal vorgelegt
 - ◆ Februar 1998 entstand der erste Entwurf
-

WWW-Konsortium (W3C)

<http://www.w3c.org>

- ◆ **ca. 150 Mitglieder (Firmen)**
 - ◆ **3 Institute: MIT Lab for Computer Science (USA), INRIA (Frankreich), Keio University (Japan)**
 - ◆ **ca. 30 hauptberufliche Angestellte**
 - ◆ **Chairman ist Tim Berners-Lee**
 - ◆ **Vermittlertätigkeit zwischen den Mitgliedern**
 - ◆ **Erstellung von Empfehlungen**
 - ◆ **Veranstalten von Treffen, Workshops und Organisation von Arbeitsgruppen**
 - ◆ **WWW-Technologieförderung in den Bereichen**
 - **Benutzerschnittstellen (HTML, CSS, PNG, VRML, SVG)**
 - **Technologie und Gesellschaft (PICS, Sicherheit, Zahlungsverkehr, Protokoll-Erweiterungen)**
 - **WWW-Architektur (HTTP, Audio, Video)**
-

Design-Ziele von XML

XML soll sich im Internet auf einfache Weise nutzen lassen

XML als SGML für das Web

- ◆ **Kompatibilität zu SGML – als eine leicht überschaubare Teilmenge ist sie somit eine Spezifikation für die Schaffung beliebig vieler Auszeichnungssprachen**
 - ◆ **Programme, die XML-Dokumente verarbeiten, sollen einfach zu schreiben sein**
 - ◆ **Unterstützung eines breiten Spektrums von Anwendungen**
 - ◆ **XML-Dokumente sollen leicht lesbar und angemessen verständlich sein**
 - ◆ **XML-Dokumente sollen leicht zu entwerfen sein**
 - ◆ **Der Entwurf von XML soll formal und präzise sein**
 - **Die Knappheit innerhalb der XML-Auszeichnungselemente ist dabei von geringerer Bedeutung – der Schwerpunkt liegt hier mehr auf der Lesbarkeit durch Maschinen als der durch Menschen**
 - ◆ **Leichter zu archivieren - Langzeitspeicherung**
 - ◆ **Wiederverwendbarkeit für die unterschiedlichsten Zwecke**
 - ◆ **Generierung der unterschiedlichsten Ausgabeformate von einer Quelle**
-

Weiterer Einsatz von XML

- ◆ Nicht mehr nur beschränkt auf die Auszeichnung von Dokumenten
 - ◆ XML generell als Mittel zum Informationsaustausch, für verteilte, entfernte Prozeduraufrufe (**RPC**, **Remote Procedure Calls**)
 - ◆ **RPC für Web Services:**
 - mit dem vom W3C vorangetriebenen **SOAP** (**S**imple **O**bject **A**ccess **P**rotocol) werden Informationen über HTTP als Transport-Protokoll auf festgelegte Weise ausgetauscht bzw. auf auf darin enthaltenen Daten reagiert
 - Die in XML kodierte Daten erlauben den Verzicht auf binäre Protokolle und sind damit verträglich für Firewalls (Port 80), durch SSL verschlüsselbar und mittels Web-Server-Farmen und Load-Balancing skalierbar
 - Web Services ermöglichen automatisierte Interaktionen zwischen Applikationen im Web
 - Auch bei Microsofts **.NET**-Strategie zum Aufbau einer Web-fähigen, integrierten Unternehmenskommunikation spielt XML die tragende Rolle
-

Extensible Markup Language

- ◆ **XML** steht für **EX**tensible **M**arkup **L**anguage
 - ◆ **Extensible:**
 - im Gegensatz zu HTML ist man nicht auf ein vereinbartes *Tagset* festgelegt, sondern es lassen sich eigene Sprachelemente kreieren
 - als eine Metasprache lassen sich mit XML sogar beliebig viele Auszeichnungssprachen generieren
 - so besteht die Möglichkeit zum Design von spezifischen Sprachen für spezielle Wissenschaftsbereiche oder Anwendungsgebiete
 - Ziel ist dabei jedoch, dass sie Aussagekraft bzgl. des Inhaltes haben
<kursthemem>, <kursort>, ...
 - Die Schöpfung neuer Tags erfordert somit ein gründliches Verständnis des Dokumentinhaltes
 - ◆ **Keine Begrenzung der Namensräume (Namespace) und in der strukturellen Komplexität**
-

Extensible Markup Language

Markup:

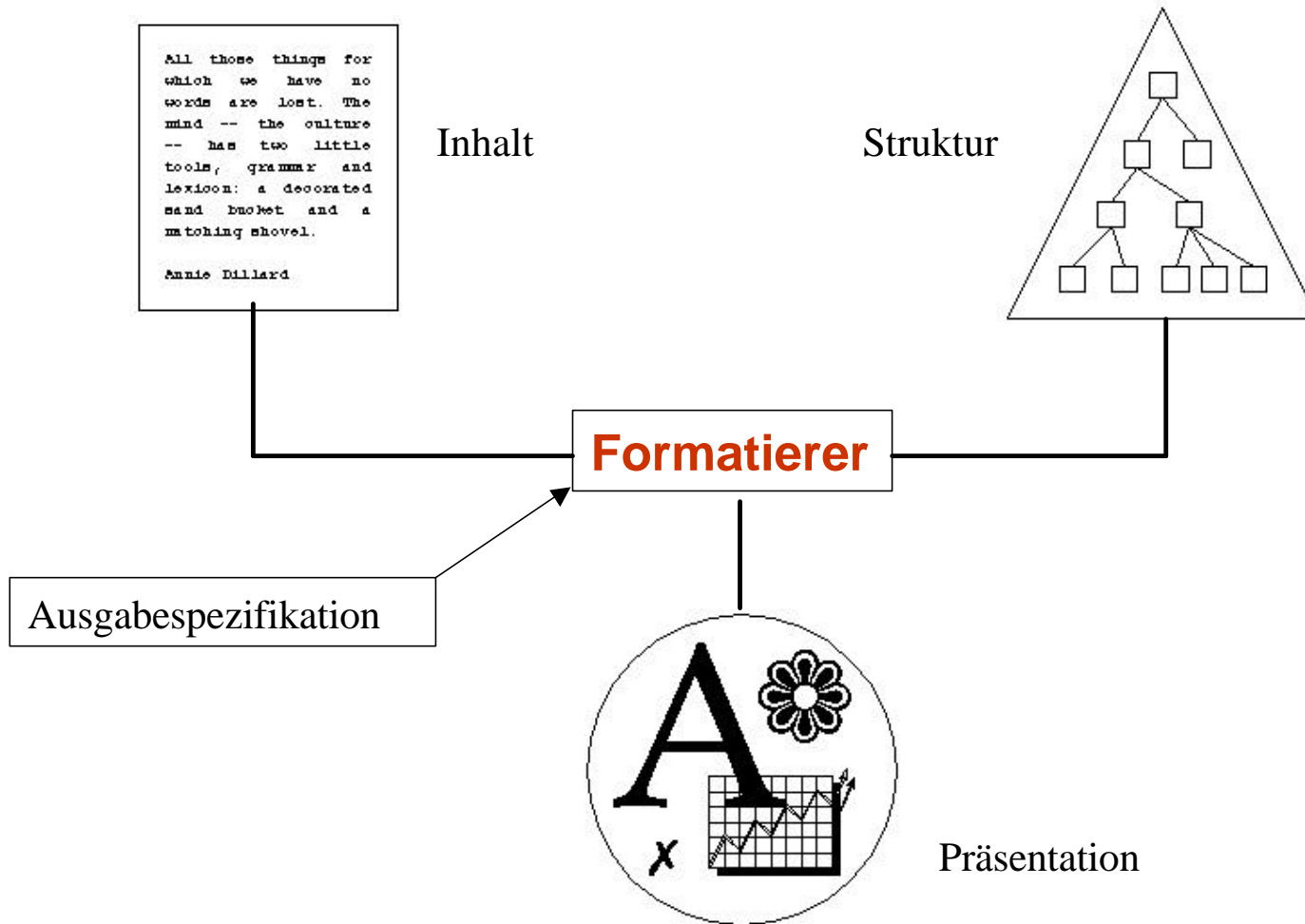
- ◆ Texte werden ausgezeichnet (Markup), um Teilen von ihnen eine Bedeutung zu geben
= Auszeichnung eines Textes durch Anweisungen
 - ◆ Nur so werden Texte für Maschinen „lesbar“
ohne diese Auszeichnung würde der Computer einen Text als eine lange Zeichenkette begreifen
 - ◆ **Procedural Markup**: Auszeichnung eines Textes mit Formatanweisungen für die **optische** Aufbereitung
 - Formatanweisungen der Text(satz)systeme wie Troff und TeX/LaTeX
 - ◆ **Generalized Markup**: Abheben auf die Dokumentenstruktur und das Ziel, die Strukturanweisungen zu formalisieren
 - SGML, XML
 - ◆ Über die Auszeichnungselemente erhalten die Textteile eine Bedeutung im Zusammenhang des Dokumentes (Semantik)
-

Extensible Markup Language

- ◆ Sprachen dienen i. allg. dazu, etwas zu beschreiben
 - ◆ Bei einer Sprache wie XML hat man es mit einem festen Umfang von Regeln zu tun
 - ◆ Auch wenn die Sprachelemente frei definierbar sind, muss doch die Syntax und Struktur klar und eindeutig festgelegt sein
 - ◆ Die Struktur wird definiert durch die Beziehungen, die jedes Element mit den anderen hat
 - ◆ Die Bedeutung der Elemente ist deutlich getrennt von der visuellen Erscheinung
 - ◆ Die Dokumentstruktur wird auch als Baumstruktur interpretiert
 - ◆ Struktur und Formatierung ist deutlich zu trennen
- Beispiel:

- **Struktur: dies ist eine Überschrift**
Formatierung: setze den Text in 24pt Helvetica
-

Konzept von XML



XML im Vergleich

Word	LaTeX	XML
Dokumentvorlage	Dokumentklasse <i>documentclass</i>	Dokumenttyp Definition DTD
Formatvorlagen	Befehle, Umgebungen	Elementtypen

Vorteile von XML

- ◆ **Basiert auf internationalen Standards**
 - **Internationalisierung durch Unicode-Unterstützung**
 - ◆ **Keine Begrenzung durch ein beschränktes Tag-Arsenal**
 - ◆ **Kontrollmöglichkeit der Gültigkeit der Dokument-Strukturen**
 - ◆ **Automatische Generierung von Navigation und Datenstrukturen**
 - ◆ **Bessere Langzeit-Archivierung**
 - ◆ **Leichte Wiederverwendung von Dokumenten:**
Formatierung auf verschiedene Ausgabemedien hin bei ein und derselben Datenquelle
 - **ermöglicht so benutzerspezifische Betrachtungsweisen**
 - ◆ **Leichtere Pflege größerer Web-Sites**
 - ◆ **Neu: dient zum Aufbau von verteilten Web-Services**
- XML beschreibt nicht, wie eine Seite aussieht und was sie tut (Aktion), sondern was jedes Wort im Text bedeutet (Semantik)**
-

XML-Dateien im Web

- ◆ XML wird HTML bei kleineren Projekten (eigene Homepages, etc.) nicht verdrängen
 - ◆ XML bietet sich im besonderen Maße zur Strukturierung größerer Web-Sites an
 - ◆ Damit ergibt sich die Anforderung, XML-Dokumente visualisieren zu können – zwei Verfahren:
 - ◆ Clientseitig: die Darstellung übernimmt ein XML-fähiger Browser
 - Über **Processing Instructions** wird festgelegt, welche Stylesheets zur Anwendung kommen (CSS oder XSL)
 - ◆ Serverseitig: Programme auf dem WWW-Server übernehmen die Umwandlung der XML-Dateien in Abhängigkeit des verwendeten Browsers
 - Über **Content Negotiation** versucht der Server herauszufinden, was dem jeweiligen Browser genehm ist
 - Der Server liefert generell HTML aus
-

Einsatz von XML heute?

- ◆ **XML soll HTML nicht ersetzen sondern überall dort ergänzen, wo die semantische Bedeutung des Datenmaterials von Bedeutung ist**
 - z. B. bei der Konvertierung, bei dem Einsatz von Datenbanken
 - ◆ **Durch die deutliche Trennung von Dokumentinhalt und deren Visualisierung benötigt XML allerdings Präsentationshilfen: CSS, XSL, DSSSL**
 - ◆ **Wie reagieren die Hersteller?**
 - Microsoft bietet XML-Unterstützung bereits im Internet Explorer 4 und deutlich verbessert im IE 6.0
 - Rudimentäre Unterstützung auch in Office 2000/2002
 - Netscape unterstützt XML erst in seiner neuen Layout-Engine „Gecko“ des **Netscape 6.1** bzw. **Mozilla**
 - Unterstützung auch in **WordPerfect 2000**, Adobe **FrameMaker 5.5.6/6.0**,
 - **KDE KOffice**, GNOME-Office: **Abiword**, **StarOffice 6.0** verwenden XML bereits als internes Datenformat
-

XML Dokument

- ◆ Eine XML-Datei besteht aus ASCII-Text
 - Je nach Encoding-Anweisung im Prolog sind auch Sonderzeichen erlaubt
 - ◆ Es setzt sich zusammen aus einem Prolog
 - Deklaration,
 - Eine optionale Angabe des Dokumenttyps bzw. der DTD, die die Strukturanweisungen vorgibt
 - und den Elementen
 - Der mit den vereinbarten Tags ausgezeichnete Inhalt
 - ◆ Die Dateiextension ist **.xml**
 - ◆ XML-Dokumente sind **wohlgeformt**, d. h. sie müssen regelkonform sein
 - ◆ Sie können darüber hinaus als Instanz einer **DTD** auch noch **gültig** sein
-

XML Dokument - Prolog

- ◆ Der Prolog enthält die XML-Deklaration, die besagt, dass es sich hier um ein XML-Dokument handelt:

`<?xml version="1.0"?>` sollte jedes XML-Dokument einleiten

- ◆ mögliche Angaben:

`<?xml version="1.0" standalone="yes" encoding="UTF-8"?>`

- ◆ **version:** derzeit aktuell ist die Version 1.0 – für den Fall, dass mehrere Versionen existieren, wird diese Angabe an Bedeutung gewinnen

`<?xml version="1.0" standalone="yes" encoding="UTF-8"?>`

- ◆ **standalone:** hiermit wird einem Programm mitgeteilt, dass es keine DTD gibt

`<?xml version="1.0" standalone="yes" encoding="UTF-8"?>`

- ◆ **encoding:** die Art und Weise, wie Sonderzeichen und Umlaute im XML-Dokument kodiert werden – der Standard ist **UTF-8**

`<?xml version="1.0" encoding="UTF-8"?>`

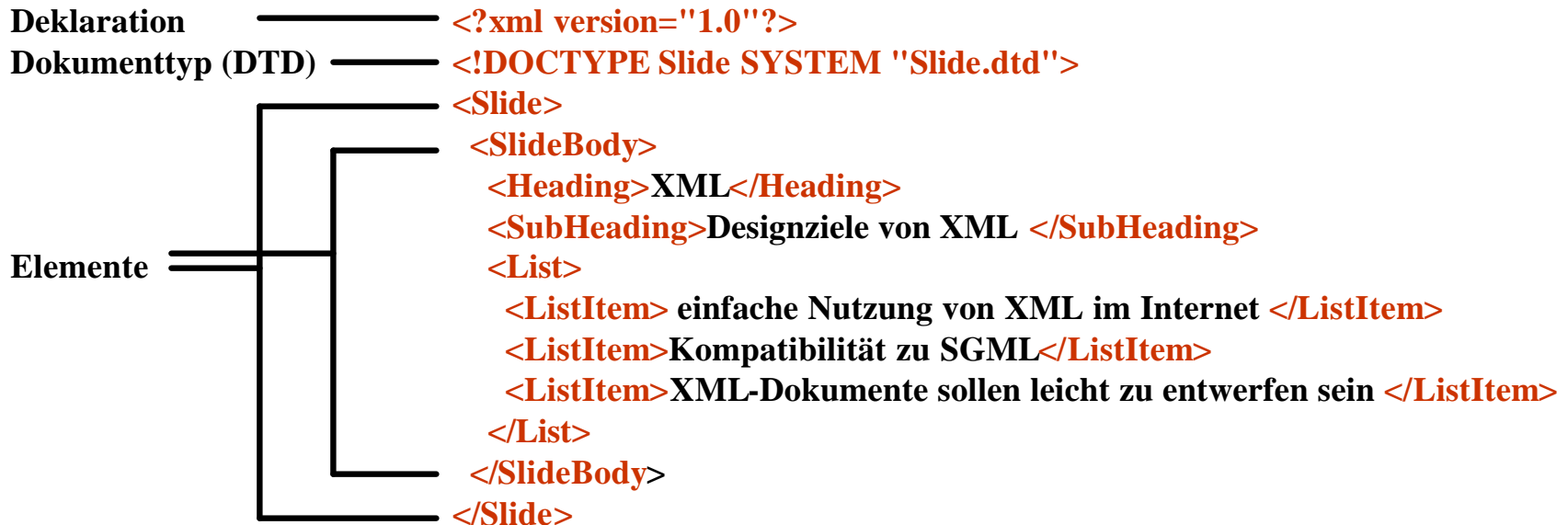
`<?xml version="1.0" encoding="ISO-8859-1"?>`

- ◆ **Optionale Angabe des Dokumenttyps:**

`<!DOCTYPE hallo SYSTEM "test.dtd">`

XML Dokument – ein Beispiel

- ◆ Beispiel eines **wohlgeformten** XML-Dokuments, das hier auch noch **gültig** ist gegenüber einer **DTD** (Slide.dtd) – d. h. es kann überprüft werden, ob die Struktur korrekt ist und die verwendeten Tags somit richtig gesetzt sind



XML Dokument - Regeln

- ◆ **Element:** sie sind die elementaren Konstrukte in XML-Dokumenten
 - Elemente werden durch die umgebenden *Tags* gekennzeichnet
`<author>Arno Amoebe</author>`
 - Wie in HTML/SGML werden auch hier die Elemente (*Tags*) in Spitzklammern gesetzt
`<list>`
 - ◆ **Attribut:** ermöglicht eine nähere Beschreibung der Elemente
 - Attribute setzen sich aus einem Namen und einem Wert zusammen
`<author kap="4">Arno Amoebe</author>`
 - jedes Attribut darf pro Element nur einmal vergeben werden
 - ◆ **Attributwert:**
 - Attributnamen und Werte werden durch Gleichheitszeichen verbunden und die Werte **müssen** in Anführungszeichen stehen
`<author kap="4">Arno Amoebe</author>`
-

XML Dokument – Regeln (2)

- ◆ **im Gegensatz zu HTML und das ihm zugrundeliegende SGML ist auf die Groß- und Kleinschreibung zu achten**
`<author>` ist nicht gleich `<Author>`
 - ◆ **Alle Tags müssen geschlossen werden - XML-Befehle sind somit stets Umgebungsbefehle**
`<author> ...Text... </author>`
 - Hier kann also nicht auf die „Gutmütigkeit“ der Browser vertraut werden
 - ◆ **auch „leere“ Tags, die keine wirkliche Umgebung spezifizieren, müssen geschlossen sein**
``
`
`
 - ◆ **Die Elemente müssen korrekt ineinander verschachtelt sein.**
Wird ein *Tag* innerhalb eines anderen geöffnet, muss er auch vor dem Ende der ihn umschließenden Umgebung geschlossen werden
`<liste> das ist der <wichtig>Aspekt</wichtig></liste>`
1 2 2 1
-

XML Dokument – Regeln (3)

- ◆ **Kommentare werden wie in HTML gesetzt**
`<!-- dies ist ein Kommentar -->`
innerhalb der Kommentare dürfen die Zeichen -- nicht auftreten
 - ◆ **Ein XML-Dokument muss mindestens ein Element enthalten**
 - ◆ **Es darf allerdings nur ein Wurzel-Element geben**
 - ◆ **Erlaubte Zeichen für Element- und Attributnamen sind:**
a - z, A – Z, 0 -9, -, _, : und Punkte innerhalb einer Zeichenkette
 - ◆ **Elementnamen dürfen nicht mit dem reservierten Wort "xml" oder "XML" beginnen**
 - ◆ **XML-Dokumente sollten stets mit einer XML-Deklaration beginnen**
-

XML Dokument – Entities

- ◆ Der von XML verwendete Zeichensatz (**ISO/IEC 10646**) deckt im allgemeinen den westeuropäische Zeichenvorrat ab (**ISO latin 1**)
 - ◆ Zeichenreferenz für Zeichen, die in XML eine besondere Bedeutung haben (Maskierung)
 - **&**; **&** (ampersand)
 - <**; **<** (less than)
 - >**; **>** (greater than)
 - '**; **'** (apostrophe)
 - "**; **"** (quotation mark)
 - ◆ oder solche, die über die Tastatur nur schwer einzugeben sind
 - **&#Zahl;** bzw. **&#xHexzahl;**
 - »** für »
 - «** für «
-

XML Dokument – weitere Regeln

- ◆ Definitionen eigener Entity-Referenzen für eine abkürzende Schreibweise, auf die man sich später beziehen kann (s. a. bei DTDs)

```
<?xml version="1.0"?>  
<!DOCTYPE buch system "buch.dtd" [ <!ENTITY kurs "Kurs: WWW  
für Fortgeschrittene" ]>
```

Verwendung dann im XML-Dokument:

```
<absatz>Im &kurs; wird folgendes zur Sprache kommen  
...</absatz>
```

- ◆ Processing Instruction (PI) erlauben Anweisungen für die Verarbeitung von XML-Dateien und werden in `<? ?>` eingeschlossen

```
<?xml version="1.0"?>  
<?xml-stylesheet href="test.xsl" type="text/xsl"?>
```

- ◆ für wörtlich übernommenen Text eignet sich die CDATA-Anweisung (Character Data)

```
<![CDATA[<absatz>das ist ein XML-</absatz>]]>
```

Wohlgeformtheit (*wellformed*)

- ◆ **Jedes XML-Dokument muss wohlgeformt (**well-formed**) sein, d. h. es muss den folgenden Regeln genügen**
 - Unterscheidung von Groß-/Kleinschreibung
 - Jeder *Start-Tag* muss auch einen korrespondierenden *Ende-Tag* haben
 - „Leere“*Tags* werden durch einen Schrägstrich **
** abgeschlossen
 - Die Elemente müssen korrekt ineinander verschachtelt werden
 - Die Attributwerte müssen in Anführungszeichen gesetzt werden
 - Sind die Zeichen **<**, **>**, **"** gemeint, dann müssen die dafür bereitgestellten Entitäten (**<**, **>**, **"**;) verwendet werden
 - Es darf nur ein Wurzelement geben
 - ◆ **Damit erst ist ein XML-Dokument – im Gegensatz zu HTML - auch für Maschinen/Programme - den sog. **Parser** - überprüfbar**
-

Gültigkeit

- ◆ Soll das Dokument darüber hinaus auch noch gültig sein, überprüft der Parser, ob die Struktur und die Anzahl der Tags gemäß einer Strukturanweisung korrekt und sinnvoll eingesetzt wurden
 - ◆ Diese Anweisungen liefert die optionale **DTD** (**D**ocument **T**ype **D**efinition)
 - ◆ Die DTD bestimmt das Regelwerk, die Grammatik für eine bestimmte in XML verfasste Auszeichnungssprache
 - ◆ Der durch die DTD festgelegte Dokumenttyp bestimmt, welche Elementtypen mit welchen Attributen in welcher Verschachtelung verwendet werden dürfen
 - ◆ XML stellt als Metasprache somit die Anleitungen zum Verfassen von DTDs zur Verfügung
-

Parser

- ◆ **Parser sind Programme, die ein XML-Dokument einlesen und auf Wohlgeformtheit überprüfen; bei nicht wohlgeformten Dokumenten wird eine entsprechende Fehlermeldung ausgegeben**
 - ◆ **Es gibt validierende und nicht-validierende Parser, je nachdem ob sie auch auf Gültigkeit prüfen**
 - ◆ **nicht-validierende Parser müssen aber immer noch Verletzungen bzgl. der Wohlgeformtheitsbeschränkung vermelden**
 - ◆ **Validierende Parser vermelden Verletzungen der Beschränkungen, die durch die Deklarationen in der DTD formuliert werden, sowie jedes Nicht-Erfüllen der in dieser Spezifikation formulierten Gültigkeitsbeschränkung**
 - **Dazu müssen sie die gesamte DTD und natürlich auch alle extern deklarierten Entitäten, auf die im Dokument verwiesen wird, einlesen und verarbeiten**
-

Parser (2)

- ◆ Grundlage aller Parser sind die **SAX**-APIs von Megginson
„SAX“ steht für **S**imple **A**PI for **X**ML
 - ◆ **Expat** von James Clark
<http://www.jclark.com/xml/expat.html>
 - Ein nicht-validierender nur auf Wohlgeformtheit prüfender Parser
 - Wird in vielen Programmen eingesetzt, u. a. in Mozilla, PHP4
 - Aufruf: **xmlwf** (Unix) bzw. **xmlwf.exe** (Windows)
 - ◆ **Java Project X** von SUN Microsystems
<http://java.sun.com/xml>
 - Komplette in Java geschriebene XML-Bibliothek, basiert auf SAX
 - Sie unterstützt die Prüfung auf Wohlgeformtheit, der Gültigkeit gegen eine DTD und bietet diverse Manipulationsmöglichkeiten bzgl. der XML-Struktur (z. B. Visualisierung)
 - Bestandteil des Java JDK 1.3
-

Parser (3)

MSXML von Microsoft

<http://msdn.microsoft.com/xml>

- ◆ **Leistungsfähiger im Internet Explorer integrierter XML-Parser**
 - MSXML 2.0 wird mit dem Internet Explorer 5.0 ausgeliefert
 - MSXML 2.5 kommt mit Windows 2000
 - ◆ **MSXML 3.0 integriert im Internet Explorer 6.0 und in Windows XP**
 - Unterstützt u. a. XSLT 1.0
 - ◆ **Installation des Updates**
 - Download und Aufruf der Datei **msxmlwr.exe**
 - Zum Update der XML-Engine **xmlinst.exe** aufrufen (unter win9x in \windows\system, unter NT in \winnt\system32), damit **msxml3.dll** Version **8.0.7820.0** zum Standard-Parser im IE wird
 - **iexmltls.exe** dient zur erweiterten Darstellung des Parsers (zusätzliches Kontextmenü) im Internet Explorer
-

Parser (4)

SP 1.3.4 von James Clark

<http://www.jclark.com/sp>

- ◆ **nsgmls** (wird auch von **PSGML** verwendet) parst und validiert SGML - und XML-Dokumente
 - Nur textbasiert! - Für UNIX und Win32 verfügbar
 - ◆ **Optionen:**
 - **-wTyp**: kontrolliert die Warnungen und Fehlermeldungen
 - **-wxml**: informiert darüber, dass es sich um eine XML-Instanz handelt ausgelegt auf die XML-spezifischen Warnungen vor Konstruktionen, die in XML nicht erlaubt sind
 - **-s**: unterdrückt die Ausgabe, Fehlermeldungen sind davon aber nicht betroffen
 - **-c Katalog**: Angabe einer Katalogdatei, im Falle von XML enthält sie nur den Verweis auf die entsprechenden Deklarationen
[xml.soc](#) mit dem Inhalt: `SGMLDECL "xml.dcl"`
-

Parser (5)

- ◆ **XML4J - XML for Java Parser v2.0/3.0** von IBM,
 - ◆ Jetzt als **Xerces** im Apache-Projekt
<http://xml.apache.org>
 - ◆ Modularer in Java geschriebener Parser
 - ◆ Validiert, generiert und manipuliert XML-Daten
 - ◆ Unterstützt die SAX-API
 - ◆ Dient als Grundlage für viele Applikationen wie auch Editoren
 - ◆ Prüfen auf Wellformness:
[xml4j.bat](#) *Datei*
 - ◆ Prüfen auf Gültigkeit:
[xml4jv.bat](#) *Datei*
-

XML-Browser: Mozilla

- ◆ Netscape's neue Layout-Engine „**Gecko**“ im Netscape 6.1/Mozilla
www.mozilla.org
- ◆ Netscape 6.1 entspricht Mozilla 0.9.2.1
- ◆ Die neueste Version Gecko/20011011 wird mit Mozilla 0.95 geliefert
- ◆ Mit Hilfe von Erweiterungen, sog. xpi-Dateien, lässt sich Mozilla zum Funktionsumfang von Netscape 6 hochrüsten
- ◆ Unterstützung von
 - HTML 4
abwärtskompatibel
zu Communicator 4
 - CSS1 und CSS2
 - XML 1.0 standardmäßig
über CSS 1/2
 - Parser: **Expat** v. J. Clark
 - XSLT über **TransforMiiX**



XML-Browser: MS Internet Explorer

◆ Microsofts Internet Explorer **6.0** (6.0.2600.0000)

◆ Unterstützt

➤ HTML 4 und proprietäre Erweiterungen

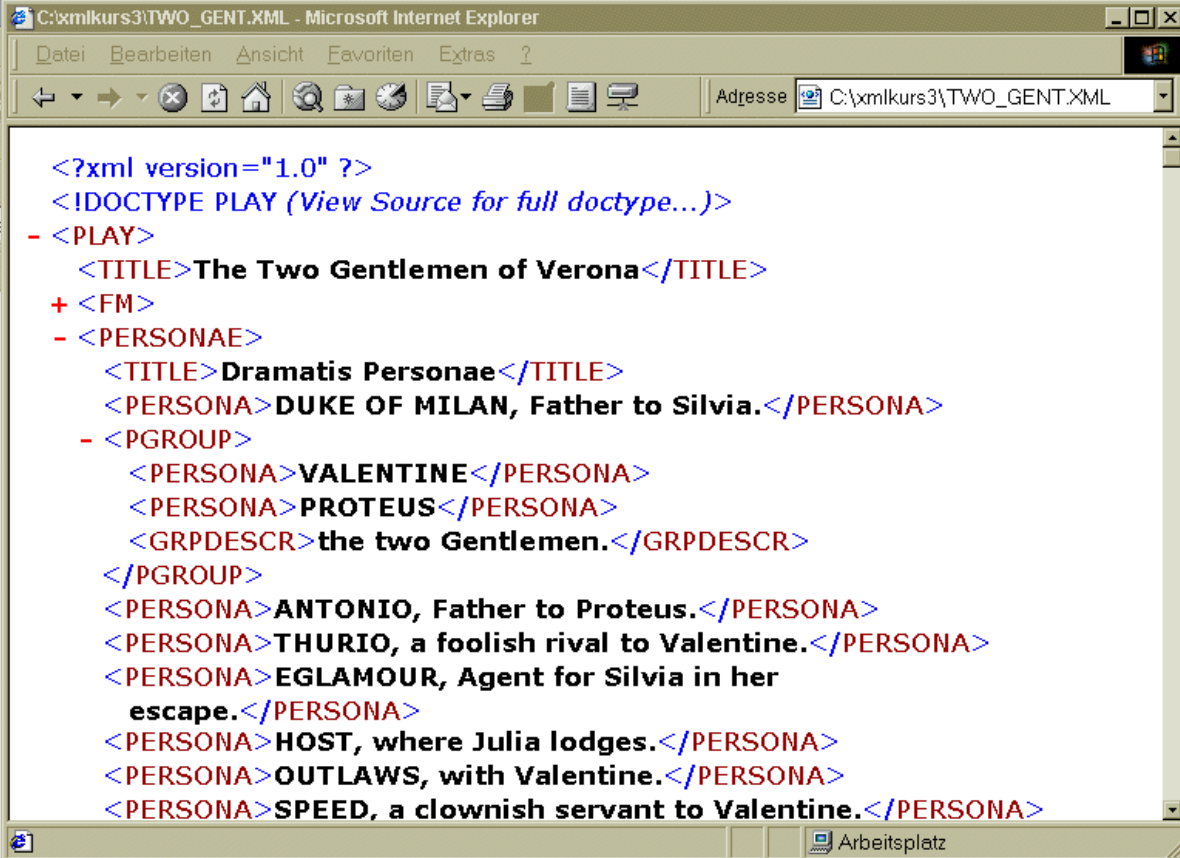
➤ CSS1 und CSS2

➤ XML 1.0 über CSS und XSL

➤ XSL/XSLT 1.0

➤ Parser:
MS-eigener
Msxml 3.0

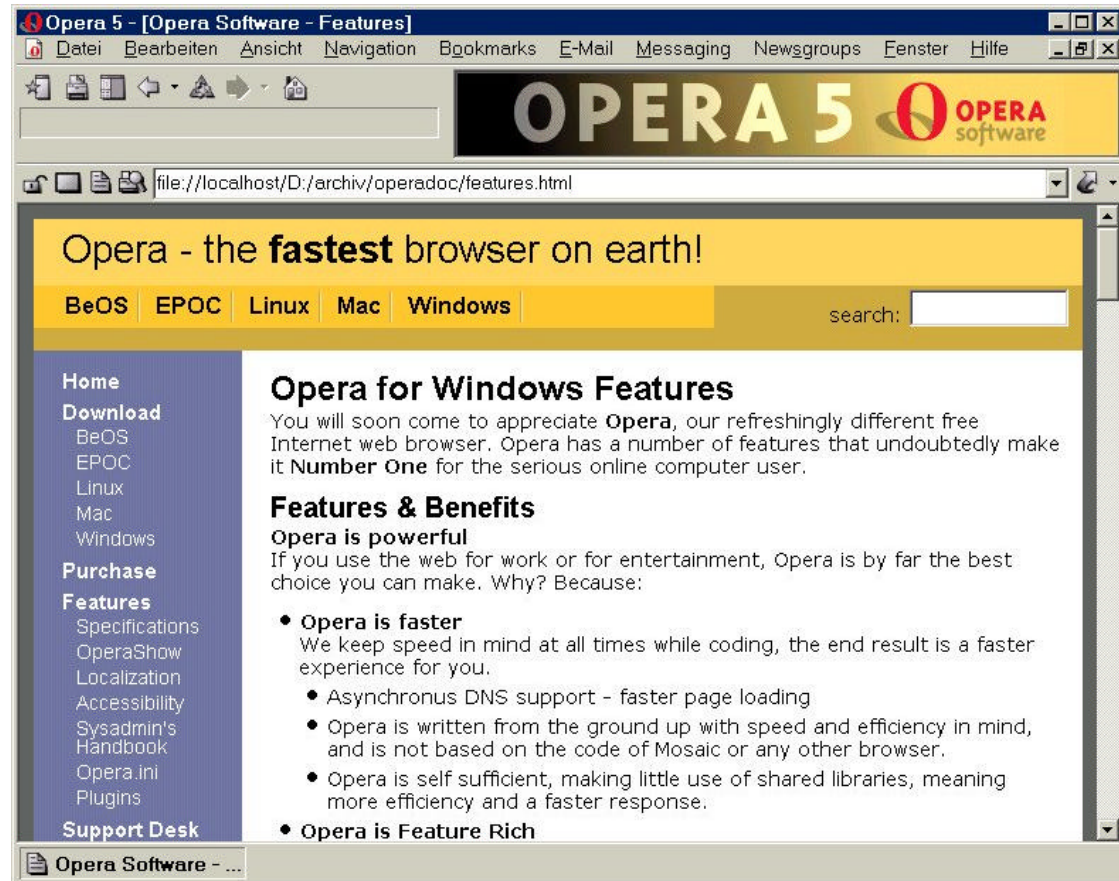
➤ Update auf
Msxml 3.0 SP1



```
<?xml version="1.0" ?>
<!DOCTYPE PLAY (View Source for full doctype...)>
- <PLAY>
  <TITLE>The Two Gentlemen of Verona</TITLE>
+ <FM>
- <PERSONAE>
  <TITLE>Dramatis Personae</TITLE>
  <PERSONA>DUKE OF MILAN, Father to Silvia.</PERSONA>
- <PGROUP>
  <PERSONA>VALENTINE</PERSONA>
  <PERSONA>PROTEUS</PERSONA>
  <GRPDESCR>the two Gentlemen.</GRPDESCR>
</PGROUP>
  <PERSONA>ANTONIO, Father to Proteus.</PERSONA>
  <PERSONA>THURIO, a foolish rival to Valentine.</PERSONA>
  <PERSONA>EGLAMOUR, Agent for Silvia in her
    escape.</PERSONA>
  <PERSONA>HOST, where Julia lodges.</PERSONA>
  <PERSONA>OUTLAWS, with Valentine.</PERSONA>
  <PERSONA>SPEED, a clownish servant to Valentine.</PERSONA>
```

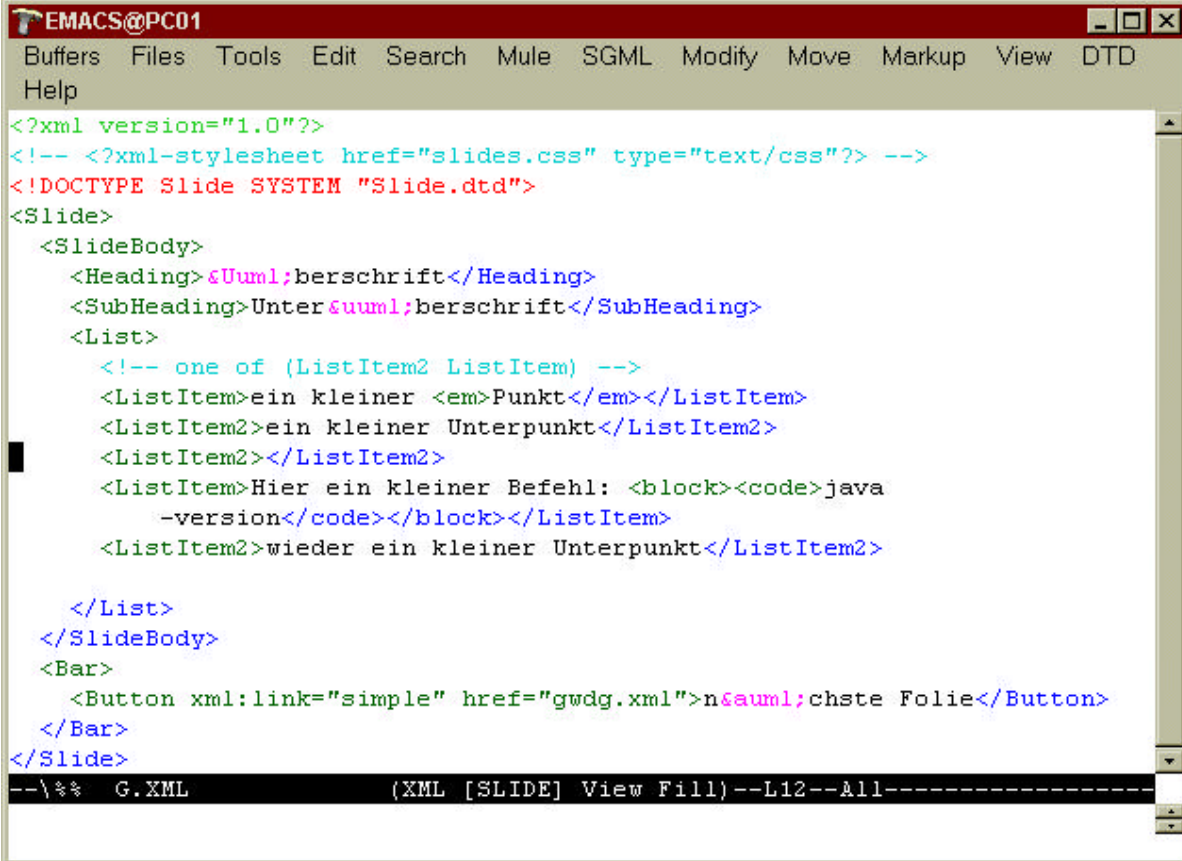
XML-Browser: Opera

- ◆ Opera 5.12 avanciert zum dritthäufigsten Browser
- ◆ Shareware - seit der Version 5 mit Werbebanner frei verfügbar
- ◆ Linux-Version, MAC-Version derzeit im Beta-Test
- ◆ Unterstützt CSS1, CSS2, XML, WML
- ◆ Darstellung von XML über CSS1 und CSS2
 - Interessanter XML-Präsentationsmodus: **Operashow**
- ◆ **Expat-Parser**



XML-Editoren

- ◆ **PSGML 1.2.2** v. *Lennart Staflin*
- ◆ **GNU Emacs ab 20.3**
und Xemacs ab 19.9
- ◆ **Kostenfrei**
- ◆ **Für Win32, Unix**
- ◆ **Integriert den Parser SP** von **J. Clark**
- ◆ **Separater DTD-Mode:**
TDTD 0.7.1
v. *Tony Graham*
- ◆ **Separater XSL-Mode**
xslide 0.1.2
v. *Tony Graham*



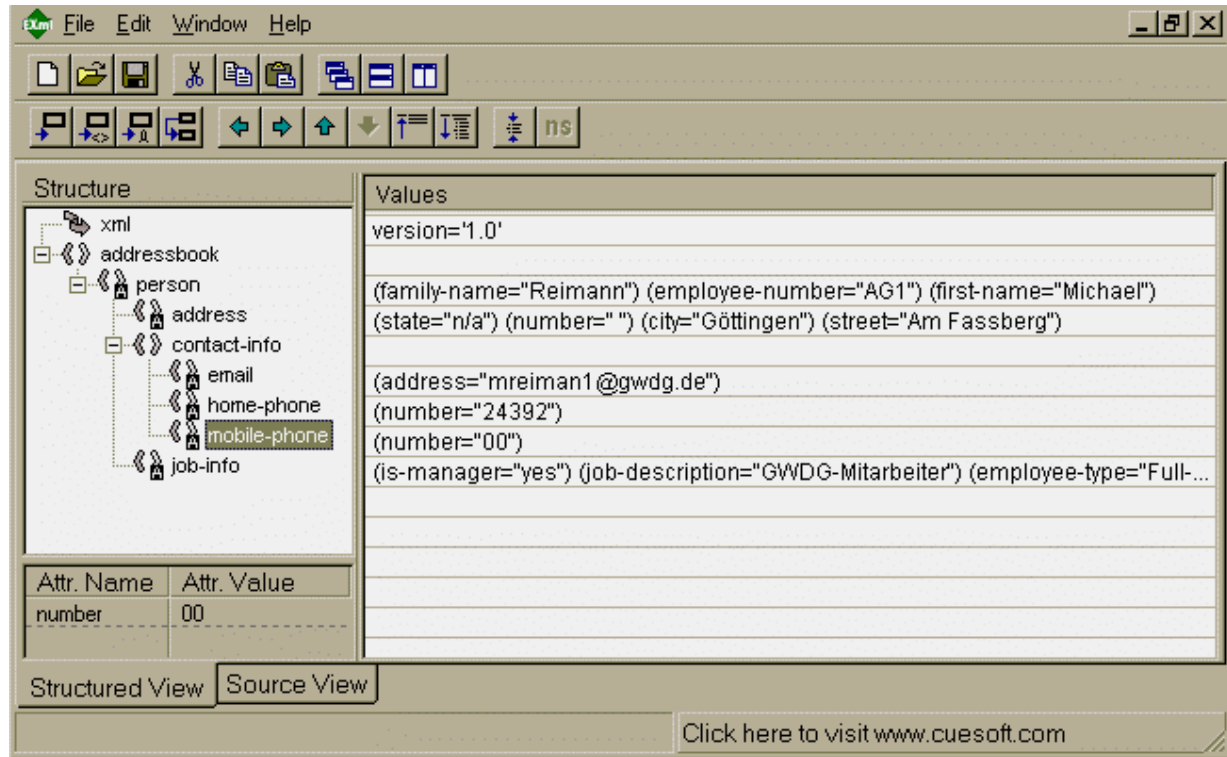
```
EMACS@PC01
Buffers Files Tools Edit Search Mule SGML Modify Move Markup View DTD
Help
<?xml version="1.0"?>
<!-- <?xml-stylesheet href="slides.css" type="text/css"? -->
<!DOCTYPE Slide SYSTEM "Slide.dtd">
<Slide>
  <SlideBody>
    <Heading>&Uuml;berschrift</Heading>
    <SubHeading>Unter &uuml;berschrift</SubHeading>
    <List>
      <!-- one of (ListItem2 ListItem) -->
      <ListItem>ein kleiner <em>Punkt</em></ListItem>
      <ListItem2>ein kleiner Unterpunkt</ListItem2>
      <ListItem2></ListItem2>
      <ListItem>Hier ein kleiner Befehl: <block><code>java
        -version</code></block></ListItem>
      <ListItem2>wieder ein kleiner Unterpunkt</ListItem2>
    </List>
  </SlideBody>
  <Bar>
    <Button xml:link="simple" href="gwdg.xml">n&uuml;chste Folie</Button>
  </Bar>
</Slide>
--\%% G.XML (XML [SLIDE] View Fill)--L12--All-----
```

Emacs als XML-IDE

- ◆ **PSGML-Mode 1.2.2** von Lennart Staflin
<http://sourceforge.net/projects/psgml>
 - ◆ **XSLT-process 2.1** von Ovidiu Predescu
<http://sourceforge.net/projects/xslt-process/>
 - Bietet eine Entwicklungsumgebung mit integriertem XSLT-Formatierer (**Xalan**) und Debugger auf Basis von Java
 - Ausgabe nach HTML und PDF
 - Ausrichtung auf die **DocBook-DTD**
 - ◆ **XAE – Authoring Environment for Emacs** von Paul Kinnucan
<http://sunsite.dk/jde/xae/>
 - Bietet ebenfalls eine Entwicklungsumgebung auf Java-Basis
 - Dokumentenerstellung derzeit noch auf **DocBook-Basis**
 - Umwandlung nach HTML über XSLT (**Saxon-Enging**)
-

EXML

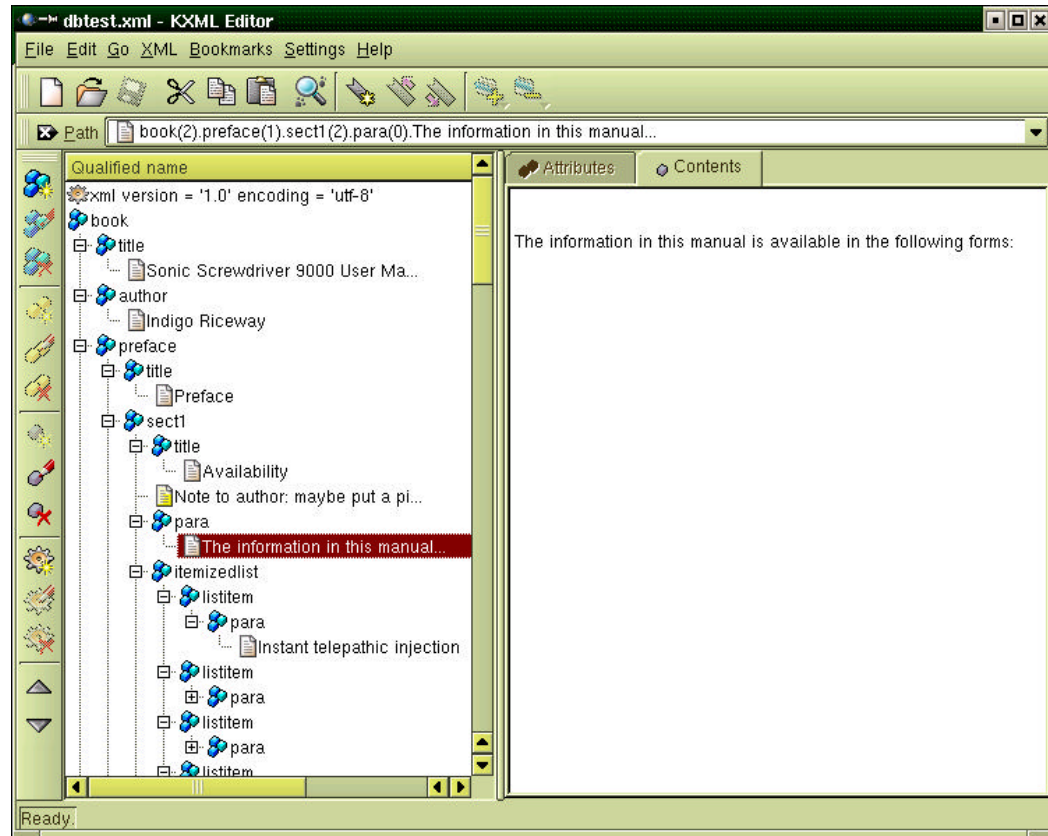
- ◆ **Exml 1.2 v. CUESoft**
- ◆ **Zeigt sowohl die hierarchische Struktur als auch den XML-Quellcode**
- ◆ **Frei verfügbar**
- ◆ **Klein und handlich**



Kxmleditor

- ◆ **Kxmleditor 0.7 v. Lumir Vanek**
- ◆ **KDE2 Linux**
- ◆ **SAX2-Parser**
- ◆ **Unterstützt die Kpart-Technologie**
- ◆ **Liest KOffice-Dateien**
- ◆ **Integration mit dem Konqueror**

kxmleditor.sourceforge.net/



Weitere Editoren

- ◆ **XML Notepad 1.5** von Microsoft
 - Kostenfreier Editor, wird nicht mehr weiterentwickelt

Kommerzielle Editoren

- ◆ **XMetaL 2.1** v. SoftQuad
 - ◆ **XMLSpy 4.0.1** v. Altova (the XML Spy Company)
 - ◆ **Adobe FrameMaker 6.0**
 - unterstützt XML durch CSS und XSL über Zusatzprodukt **WebWorks-Publisher** (Standard)
 - Verfügbar unter Win32, MAC, UNIX (Solaris, etc., nicht Linux)
 - ◆ **Corel WordPerfect 2000**
 - Windows und Linux
 - ◆ **SUN StarOffice 6.0** / baugleich mit **OpenOffice**
 - Dateiformat XML im gepackten ZIP-Format
 - ◆ **MS Office 2000/2002** – über die Funktion „*Als Webseite speichern...*“
 - ◆ **AbiWord 0.9.2** (Linux und Windows), **KOffice 1.1** (Linux)
-

WAP/WML

- ◆ Das **Wireless Application Protocol (WAP)** ermöglicht das Abrufen von Informationen aus dem Internet über ein mobiles Devices, z. B. Handy, PDA, etc.
 - ◆ derzeit über das **GSM-Netz (Global System of Mobile Communication)**
 - ◆ Bestandteil dieses Protokolls ist die auf XML aufbauende Sprache **WML (Wireless Markup Language)**, die die Möglichkeit zur Darstellung von Text und Grafik liefert, und die dazugehörige Scriptsprache **WMLScript**
 - ◆ die WML-Spezifikation (derzeit 1.1) definiert lediglich die Syntax sog. WML-Dokumente, die Darstellung bleibt den Micro-Browsern in den Handys überlassen
 - ◆ In Anbetracht der geringen Größe der Displays (z. B. Nokia 7110 mit 96x65 Pixel, ca. 4-6 Textzeilen) sollte sowohl Umfang als auch Erscheinungsbild entsprechend angepasst werden
 - ◆ brauchbare WAP-Handys: z. B. Nokia 7110, 6210, Siemens S35i
 - ◆ Die non-profit Organisation WAP-Forum sorgt für die Standardisierung
<http://www.wapforum.com>
 - ◆ Testumgebung: Nokia WAP Toolkit 1.2
<http://www.forum.nokia.com/developers/wap/wap.html>
 - ◆ Phone.com Software Developer's Kit (UP.SDK)
<http://www.phone.com/developers/index.html>
 - ◆ WAP-Funktionalität auch im Web-Browser **Opera 5.02**
-

XML-Dokument als gültige Instanz

- ◆ **DTD innerhalb des Dokumentes**

```
<?xml version="1.0"?>
<!DOCTYPE hallo [
<!ELEMENT hallo (#PCDATA)>
<!-- intern --> ]>
<hallo>Hallo Kurs!</hallo>
```

- ◆ **Dokument und DTD getrennt**

- **DTD (*hallo.dtd*)**

```
<!ELEMENT hallo (#PCDATA)>
```

- **XML-Dokument als Instanz**

```
<?xml version="1.0"?>
<!DOCTYPE hallo SYSTEM "hallo.dtd">
<!-- extern -->
<hallo>Hallo Kurs!</hallo>
```

DTD (1)

- ◆ Eine DTD ist eine Ansammlung von Regeln, die festlegen,
 - Welche *Elemente* in einem Dokument verwendet werden dürfen, welche *Inhalte* sie haben und wie sie sich aufeinander beziehen (Baumstruktur),
 - Welche *Attribute* für ein Element erlaubt sind und welche *Werte* sie annehmen dürfen
 - ◆ Eine DTD beschreibt die Struktur eines Schriftstücks und aller gleich aufgebauten Schriftstücke
 - ◆ **DTD** steht für **D**ocument **T**ype **D**efinition und darf nicht verwechselt werden mit der **Dokumenttyp Deklaration** zu Beginn einer XML-Instanz
 - ◆ In Verbindung mit Parsern und XML-Editoren erlaubt es eine DTD zu kontrollieren, ob Autoren sich an die strukturellen Vorgaben eines Texttyps, z.B. eines Artikels für eine Zeitschrift, halten. Sie wirkt unterstützend und normierend
 - ◆ Eine DTD ist zu einem XML-Dokument nicht zwingend notwendig.
-

DTD (2)

- ◆ **Soll ein XML-Dokument gültig sein, dann wird es gegen seine Strukturanweisungen (DTD) geprüft, d. h., es wird nicht nur getestet, ob es regelkonform ist, sondern ob auch alle Tags richtig gesetzt sind.**
 - ◆ **Wann braucht man eine DTD?**
 - **Das Dokument ist Teil einer größeren Dokumentsammlung, die alle die gleiche Struktur haben sollen**
 - **Es soll maschinell überprüft werden können, ob eine bestimmte Anzahl von Daten aufgeführt ist**
 - **Es soll generell maschinell verarbeitbar sein, so daß es mit anderen Dokumenten verglichen und einheitlich in andere Datenformate (z. B. Datenbanken) umgesetzt werden kann**
-

DTD (3)

- ◆ **Bevor eine DTD erstellt wird, sollte zuerst eruiert werden, ob es nicht schon eine solche oder ähnliche gibt**
 - **Docbook-DTD** von Oasis, oft für technisch-wissenschaftliche Texte eingesetzt, wurde von Norm Walsh nach XML konvertiert und mit Stylesheets versehen
 - Eine auf XML basierende HTML-Version: **XHTML**
 - ◆ **Eine selbsterstellte DTD muss nicht notwendig umfangreich werden**
 - ◆ **Für komplexe Dokumentationen kann das Design einer optimalen DTD eine längere Zeit in Anspruch nehmen**
 - ◆ **Interne DTD:** üblicherweise werden die DTDs in separaten Dateien (mit der Endung **.dtd**) angelegt
sofern die Anweisungen im XML-Dokument selber stehen, spricht man von einer internen DTD
-

DTD (4)

- ◆ Die Elementtyp-Deklaration definiert ein Element durch einen Namen und den ihm zugewiesenen Inhalt

```
<!ELEMENT name Inhalt >
```

- ◆ Aus dem Namen, den spitzen Klammern und dem Schrägstrich werden dann Start-Tag `<name>` und End-Tag `</name>` gebildet, die im XML-Dokument das Element eingrenzen.

- ◆ Einige Beispiele

```
<!ELEMENT adresse ((firma | person), strasse, ort)>  
<!ELEMENT firma (firmenname)>  
<!ELEMENT person (anrede?, vorname, nachname)>  
<!ELEMENT adressbuch (adresse)+>
```

DTD (5)

- ◆ Die *Namen* (von Elementen oder Attributen) können mit einem Buchstaben oder dem Unterstrich beginnen
 - ◆ Auch ein Doppelpunkt ist am Anfang möglich; doch sollten Doppelpunkte nur im Zusammenhang mit Namespaces verwendet werden
 - ◆ Als weitere Zeichen sind außer Buchstaben und Unterstrich noch Ziffern, der Punkt und der Bindestrich erlaubt
 - ◆ Ein Unterschied zwischen Groß- und Kleinschreibung erzeugt unterschiedliche Namen
 - ◆ Der im Zusammenhang mit Attributwerten verwendete Datentyp **NMTOKEN** ist ähnlich eingeschränkt wie ein Name
-

DTD (6)

- ◆ **Das *Inhaltsmodell* gibt an, welche untergeordneten Elemente in einem Element in welcher Kombination erlaubt sind**
 - ◆ `<!ELEMENT eintrag (lemma, wa, bd)>`
Alle Elemente, die durch das Komma als Sequenz-Operator verbunden sind, müssen vorkommen, und zwar in der Reihenfolge, in der die entsprechenden Namen auftreten
`<!ELEMENT lemma (#PCDATA)>`
Der Inhalt besteht nur aus Zeichen(ketten): Parsed Character Data
 - ◆ `<!ELEMENT bd (txt, wa?)>`
Das Fragezeichen (?) signalisiert Optionalität, das betreffende Element kann fehlen
 - ◆ `<!ELEMENT wb (eintrag)+ >`
Das Pluszeichen (+) besagt: das voranstehende Element **muss einmal, kann mehrfach vorkommen**
-

DTD (7)

- ◆ `<!ELEMENT eintrag (wort,Idiom*) >`
Der Stern-Operator (*) erlaubt, dass das vorangehende Element sowohl fehlen als auch mehrfach vorkommen darf
 - ◆ `<!ELEMENT head (access | meta) >`
Hier wird der ODER-Operator (|) verwendet: Nur eines der beiden Elemente darf benutzt werden
 - ◆ `<!ELEMENT head (access | meta)+ >`
Der ODER-Operator (|) in Kombination mit +: Eines der beiden Elemente muss auftreten, ansonsten können beide Elemente beliebig oft und in beliebiger Reihenfolge vorkommen
 - ◆ `<!ELEMENT idiomA ((txt,lmref,txt?) | (lmref,txt))>`
Treten sowohl Sequenz- als auch ODER-Operator in einem Inhaltsmodell auf, müssen ihre Gültigkeitsbereiche durch Klammern abgegrenzt werden
-

DTD (8)

- ◆ **Kombinierte Inhaltsmodelle i. e. S.** enthalten sowohl PCDATA als auch Elementnamen.
 - ◆ `<!ELEMENT absatz (#PCDATA | em | foreign)* >`
Bei der Mischung von Text und weiteren Elementen soll #PCDATA am Anfang stehen. Reihenfolgebeziehungen können nicht festgelegt werden. Es ist nur die Operatorkombination wie im Beispiel erlaubt
 - ◆ `<!ELEMENT hr EMPTY >`
EMPTY deklariert ein leeres Element, wie etwa der **hr** in HTML
 - ◆ `<!ELEMENT alles-in-einem ANY >`
ANY erlaubt gemischte Inhalte mit *jedem beliebigen* Element, das in der DTD deklariert ist.
-

DTD (9)

- ◆ welche DTD dem jeweiligen XML-Dokument zugrunde liegt, wird durch eine **Document Type Declaration** vermittelt

```
<!DOCTYPE wb SYSTEM "/usr/global/sgml/diz5.dtd">
```

Das Wurzelement im Dokument ist `<wb>` und die Regeln befinden sich in der genannten Datei

- ◆ Document Type Declaration mit öffentlichem Bezeichner:

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"/usr/global/sgml/wml_1_1.dtd">
```

- ◆ Document Type Declaration mit (partiell) interner DTD:

```
<!DOCTYPE wb SYSTEM "diz2.dtd"[
<!ELEMENT bedeutet      (kontext?, bd+)>
<!ATTLIST bedeutet seq NMTOKEN #IMPLIED>
<!ELEMENT eintrag      (lemma, wa, bedeutet+)>
<!ELEMENT kontext      (#PCDATA)> ]>
```

Interne Definitionen haben Vorrang!

DTD (10)

- ◆ Attribute von Elementen enthalten Informationen *zum* Text, die nicht *im* Text erscheinen
- ◆ Deklaration der Attributlisten:
hier wird festgelegt, welche Attribute für einen Elementtyp existieren, von welchem Typ die Attributwerte sind, und ob sie einen Vorgabe-Wert enthalten

```
<!ATTLIST img
    src      CDATA #REQUIRED
    alt      CDATA #REQUIRED
    height   CDATA #IMPLIED
    width    CDATA #IMPLIED >
```

- ◆ Es gibt notwendige (**required**) und implizite (**implied**) Attribute. Bei den impliziten Attributen müssen die Werte nicht explizit angegeben werden. Alle vier Attribute erwarten eine Zeichenkette (**CDATA**)
-

DTD (11)

```
<!ATTLIST h1
    id ID #IMPLIED
    align (left|center|right|justify) "left" >
```

- ◆ **id** dient dazu, eine eindeutige *Identifikation* an einem Element anzubringen, damit darauf mit einem Attribut vom Typ **idref** verwiesen werden kann
 - ◆ **align** ist vom Typ Aufzählungstyp, dessen Wertebereich *vollständig* in den Klammern enthalten ist. Die Vorgabe ist hier der Wert **left**.
-

DTD (12)

- ◆ Eine **Entity** (Einheit) ist eine Ansammlung von Zeichen, auf die man als Ganzes Bezug nehmen kann
 - ◆ Allgemeine Entities werden in der DTD vereinbart und in einer Dokumentinstanz benutzt
 - ◆ Vereinbarung: `<!ENTITY Aring "Ä">`
Dabei wird eine Zeichenreferenz verwendet
 - ◆ Benutzung: `Åland` `<!-- ergibt: Åland -->`
 - ◆ `<!ENTITY FHDW "Fachhochschule für
die Wirtschaft">` `<! -- Zeichenkette als Entity -->`
-

DTD (13)

- ◆ Parameter-Entities haben die gleiche Funktionalität wie allgemeine Entities, nur ist ihr Gebrauch auf die DTD beschränkt

- ◆ Für die Parameter Entity Reference (**PEReference**) dient ein Prozentzeichen (%) als Präfix

```
<!ENTITY % block "absatz | listing | abbildung | kasten">  
<!ELEMENT buch (kapitel+)>  
<!ELEMENT kapitel (ueberschrift, (%block;)*, abschnitt+)>
```

- ◆ Dies ermöglicht eine verkürzte Schreibweise, die leichtere Wartbarkeit einer DTD und erhöht auch die Lesbarkeit
- ◆ Externe Entities, um Elementtypen, die in *mehreren* DTDs enthalten sind, nicht in jede DTD-Datei einzeln hineinschreiben zu müssen

```
<!ENTITY % tab-modell SYSTEM "/usr/local/sgml/tabelle.dtd">  
%tab-modell;  
<!ENTITY % block "absatz | ... | kasten | tabelle">
```

- ◆ die Tabellen-Definition im Block **block** wurde hier aus der externen Datei **tabelle.dtd** entnommen, sodann expandiert (**%tab-modell;**) damit ist die Deklaration des Elementtyps **tabelle** verfügbar
-

DTD (14)

- ◆ **Zusätzlich zu den systemabhängigen Bezeichnern gibt es auch noch die **Public Identifier**. Sie sind system-unabhängig und müssen vom XML-Prozessor in geeigneter Weise in einen Dateinamen o.ä. umgewandelt werden**

```
<!ENTITY % HTMLlat1 PUBLIC  
"-//W3C//ENTITIES Latin1//EN//HTML"  
"http://www.w3.org/DTD/ISOlat1.ent">  
%HTMLlat1;
```

Hier wurden die Entities für den ISO-Latin-1-Zeichensatz mit einer Kombination aus Public und System Identifier geladen

Beispiel

- ◆ **Eine kleine DTD für E-Mails.**

Im Wesentlichen besteht eine E-Mail aus einem Empfänger, einem Absender, einem Thema (Betreff oder »Subject«) und der eigentlichen Nachricht

```
<!-- E-Mail-DTD Version 1 -->
<!ELEMENT email (empfaenger, absender, thema, nachricht)>
<!ELEMENT empfaenger (#PCDATA)>
<!ELEMENT absender (#PCDATA)>
<!ELEMENT thema (#PCDATA)>
<!ELEMENT nachricht (#PCDATA)>
```

➤ **PCDATA** steht für den Text (Parsed Character Data)

- ◆ **Beispiel eines XML-Dokumentes**

```
<?xml version="1.0"?>
<!DOCTYPE email SYSTEM "email1.dtd">
<email>
  <empfaenger>Marta Mikrobe</empfaenger>
  <absender>Arno Amoebe</absender>
  <thema>XML-Kurs</thema>
  <nachricht>der Kurs beginnt Dienstag um 9.15 </nachricht>
</email>
```

Beispiel (2)

◆ Erweiterung der DTD für E-Mails.

```
<!-- E-Mail-DTD Version 1 -->
<!ELEMENT email (empfaenger, kopieAn*, absender, thema,
nachricht)>
<!ELEMENT empfaenger (#PCDATA)>
<!ELEMENT kopieAn (#PCDATA)>
<!ELEMENT absender (#PCDATA)>
<!ELEMENT thema (#PCDATA)>
<!ELEMENT nachricht (#PCDATA)>
```

➤ **PCDATA** steht für den Text (Parsed Character Data)

◆ Beispiel eines XML-Dokumentes

```
<?xml version="1.0"?>
<!DOCTYPE email SYSTEM "email1.dtd">
<email>
<empfaenger>Marta Microbe</empfaenger>
<absender>Arno Amoebe</absender>
<kopieAn>Kollegen</kopieAn>
<kopieAn>XML-Liste</kopieAn>
<thema>XML-Kurs</thema>
<nachricht>der Kurs beginnt Dienstag um 9.15 </nachricht>
</email>
```

Multimedia

- ◆ Einbindung von Grafiken über eine Entity-Deklaration

```
<!ENTITY kurs_fhdw SYSTEM "fhdw/kursfoto.JPEG" NDATA JPEG>
```

Benutzung dann in der Instanz:

```
&kurs_fhdw;
```

ndata steht für **non-XML-Data**, Daten in einer bestimmten Notation (hier JPEG)

- ◆ die **Notation**-Deklaration für derartige Formate (JPEG), sollte in der DTD stehen, damit sie in jeder Instanz zur Verfügung steht

```
<!NOTATION JPEG PUBLIC "ISO/IEC 10918:1993//NOTATION  
Digital Compression and Coding of Continuous-tone Still  
Images (JPEG)//EN">
```

- ◆ Der Name des **Public Identifier** ist in diesem Fall der internationale Standard

- ◆ Alternativ dazu gibt es auch **System Identifier**, der beispielsweise den Dateinamen eines Programms zum Anzeigen der Inhalte enthält

```
<!NOTATION WAV SYSTEM "C:\windows\player.exe" >
```

DocBook DTD

- ◆ DocBook wurde anfang der 90er Jahren von O'Reilly entwickelt
 - ◆ Diente als SGML Sprachdefinition ursprünglich zur Unix-Dokumentation (*man-Pages*), wurde aber auch bald für die technisch-wissenschaftliche Dokumentation eingesetzt
 - ◆ Zwei Arten von Dokumenten: Artikel und Bücher
 - ◆ Inzwischen ist DocBook in einer XML-Version verfügbar (Version 4.1.2)
 - ◆ mittlerweile wird es für viele Dokumentationsprojekte zu freier Software eingesetzt: FreeBSD, Gnome, Darwin, KDE, etc.
 - ◆ Liefert in der SGML-Variante über **DSSSL / Jade** diverse Ausgabeformate: HTML, RTF, TeX, PDF über TeX TeXinfo, man-Pages
 - ◆ Für die XML-Version bietet sich die Formatierung über **XSLT** hin nach HTML, über **XSLT/FO** nach PDF und über **dblatex** nach LaTeX
 - ◆ Als Eingabe-Editor bietet sich der Emacs/XEmacs mit dem PSGML-Mode an
 - ◆ **Abiword** bietet eine Import/Export-Funktion für DocBook
 - ◆ **FrameMaker SGML** setzt ebenfalls auf DocBook auf
-

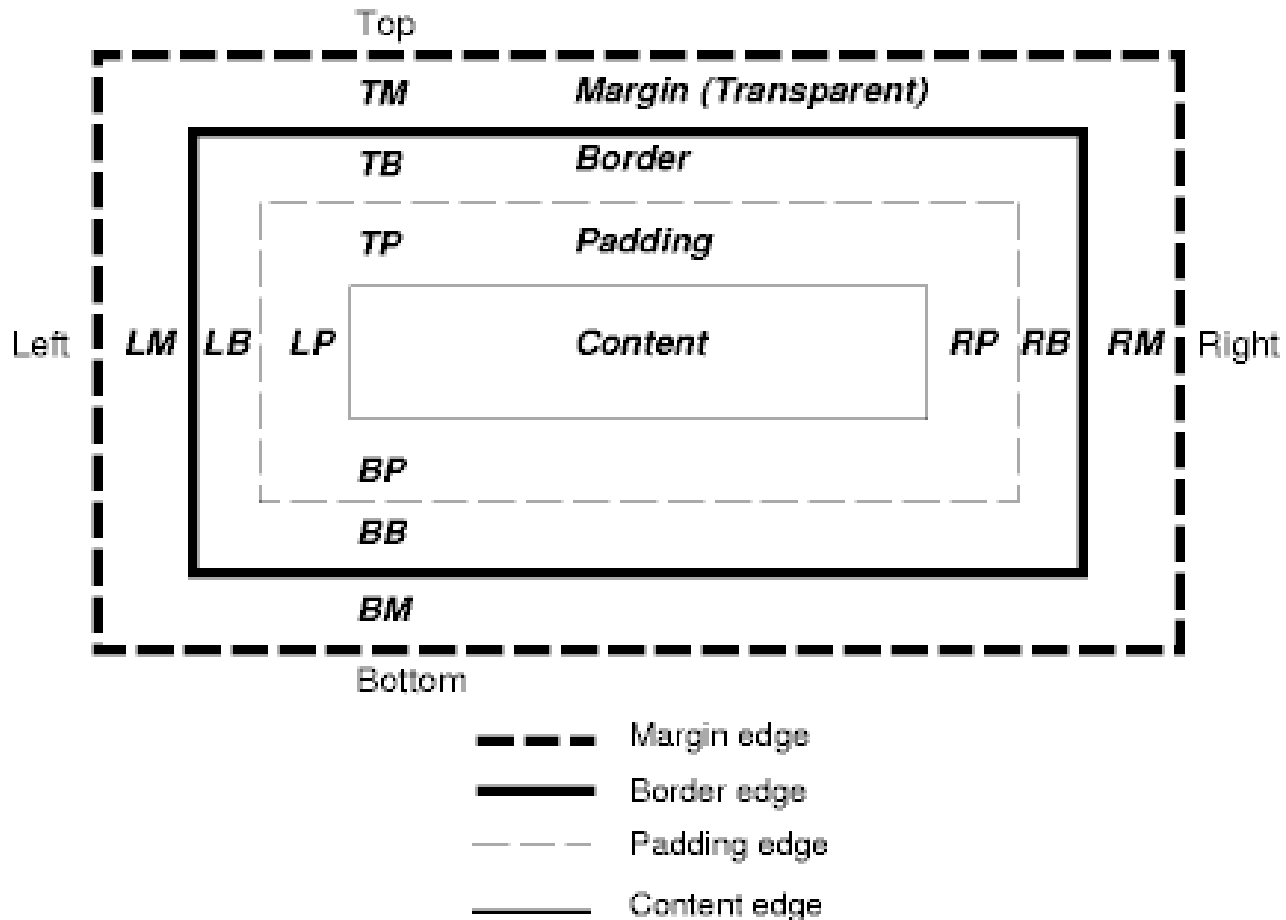
Ausgabemöglichkeiten

- ◆ **Präsentation des Datenmaterials wird **nicht** von XML geleistet – deutliche Trennung von Inhalt und Form**
 - ◆ **Da XML Informationen über das Dokument liefert, bietet es ideale Voraussetzungen zur Möglichkeit der unterschiedlichen visuellen Aufbereitung des Inhaltes in Abhängigkeit zu seiner Semantik und der damit verbundenen adäquaten Ausgabe**
 - **Für verschiedene Sichtweisen des einen Datenmaterials**
 - **Für unterschiedliche Ausgabegeräte (Drucker, PDA, etc.)**
 - ◆ **Browser können nicht wissen, wie die einzelnen Elemente repräsentiert werden - zwei Lösungsmöglichkeiten**
 - ◆ **Wandlung nach HTML über Programme wie Java, Python, Perl, etc.**
 - ◆ **Einsatz von Stilvorlagen**
 - **CSS – Cascading StyleSheets**
 - **XSL – Extensible Stylesheet Language**
 - **DSSSL – Document Style Semantics and Specification Language**
-

CSS

- ◆ **CSS bietet eine Zuordnung von Präsentationsanweisungen zu den Bereichen und Objekten einer Seite**
 - ◆ **Vorteile**
 - **Leichtverständliche Syntax**
 - **Hoher Bekanntheitsgrad durch die Verwendung mit HTML**
 - **Wird häufig für einfachere XML-Dokumente bei den Browsern eingesetzt (IE6, Mozilla, Opera)**
 - ◆ **Nachteile**
 - **Einfaches Box-orientiertes Formatierungsmodell, welches kein aufwendiges Layout erlaubt**
 - **Berücksichtigt keine Verwandtschaftsbeziehungen (kann z. B. nicht alle Abschnitte mit gleichen Nummern auch gleich formatieren)**
 - **Keine Programmierstrukturen**
 - **Wenig Möglichkeiten zur Parametrisierung der Stylesheets**
 - **Kann keinen Text generieren (Seitennummern)**
-

CSS Box-Modell



CSS (1)

- ◆ **Wachsende Akzeptanz seit der Vorstellung 1996**
 - **MS Internet Explorer ab 4.0**
 - **Netscape Communicator ab 4**
 - **Amaya (X11) - Erprobungssoftware des W3C**
 - **Opera**
 - **Netscape 6.1 / Mozilla**
 - ◆ **die derzeit verfügbaren modernen Versionen der HTML-Editoren unterstützen ebenfalls das Erstellen von CSS-Anweisungen**
 - **MS Frontpage/Office 2000, Dreamweaver 3/4, Golive 4/5, StarOffice**
 - ◆ **in einer eigenen Datei oder innerhalb eines HTML-Dokuments werden dem Browser typographische Anweisungen für bestimmte Auszeichnungselemente übergeben (z. B. Schrifttype, Schriftschnitt, Größe, Farbe, Einzug und Durchschuss)**
-

CSS (2)

- ◆ **CSS erlaubt es, Formatierung vom Basisdokument zu trennen und eine Stilvorlage für beliebig viele (HTML-)Dokumente zu nutzen**
 - ◆ **Es orientiert sich an die in der DTD (z. B. HTML) festgelegten Elemente und diese Stilanweisungen befinden darüber, welche visuelle Ausprägung diese Elemente haben sollen**
 - ◆ **Für XML sind die Stilvorlagen zwingend notwendig, da die Browser die neuen Sprachelemente gar nicht kennen können und dementsprechend auch nicht mit Formaten vorbesetzt haben**
 - ◆ **Dem Browser muss also mit der XML- auch mindestens eine CSS-Datei mitgeschickt werden**
 - ◆ **Da sich die Cascading Style Sheets auf das reine Formatieren von Markup (H1, div etc.) beschränken und keine programmieretechnischen Konstrukte erlauben gehört die Zukunft **DSSSL** und **XSL****
 - ◆ **MS **Internet Explorer**, **Opera** und **Mozilla** unterstützen CSS für die Formatierung von XML-Dokumenten**
-

CSS (3)

- ◆ Zuordnung zwischen HTML-Element und dessen Präsentation auf dem Ausgabemedium

```
H1 { color: blue }
```

Alles was in der <H1>-Umgebung steht, wird blau eingefärbt

- ◆ Die Syntax sieht wie folgt aus:

```
Selektor {Eigenschaft: Wert}
```

- ◆ die zulässigen Selektoren werden durch die DTD bestimmt

- ◆ Gruppierung von Selektoren durch Kommata getrennt

```
H1, H2, H3 { color: blue }
```

- ◆ Angabe mehrerer Eigenschaften durch Semikola getrennt

```
H1 { color: blue; font-size: 16pt;  
      font-weight: bold }
```

- ◆ unabhängige Formate: Zur Unterscheidung von Elementselektoren erhalten ID-Selektoren das Präfix #

```
#gelb { color: yellow }
```

im XML-Dokument:

```
<text ID="gelb"> .... </text>
```

CSS (4)

- ◆ **Möglichkeit zur Verschachtelung durch kontextsensitive Zuweisung**

```
ol      { list-style: decimal }
ol ol   { list-style: lower-roman }
ol ol ol { list-style: lower-alpha }
```

- **Mit dieser Definition erhalten die Aufzählungen eine jeweils andere Nummerierung**

- ◆ **Vererbung: HTML-Elemente vererben ihre Eigenschaftswerte an die eingebetteten Elemente**
- ◆ **CSS unterscheidet Textblöcke und Zeichenelemente (Box-Modell)**
- ◆ **Da der Browser bei einem XML-Dokument nicht wissen kann, mit welcher Art von Elementen er es zu tun hat, muss hier eine CSS-Anweisung helfen**

bei Textblöcken

```
text { display: block }
```

bei Zeichenelementen

```
zeichen { display: inline }
```

CSS (5)

- ◆ Da dem Browser bei ihm unbekannten XML-Elementen keine impliziten Wertvorgaben existieren, müssen die CSS-Anweisungen ausführlicher ausfallen

- ◆ Deklaration der Stilanweisungen in der XML-Datei

```
<?xml version="1.0"?>
```

```
<?xml-stylesheet type="text/css" href="kurs.css"?>
```

- ◆ unter Einbeziehung von Namensräumen ist es den Autoren von XML-Dokumenten erlaubt, Elemente aus fremden DTDs zu verwenden, die öffentlich zur Verfügung stehen

```
<html:img src="kursraum.jpg"/>
```

- ◆ Dazu sollte im Hauptelement diese externe DTD bekannt gemacht werden

```
<element xmlns:html="http://www.w3.org/TR/REC-html40">
```

hier werden Elemente aus der HTML-DTD verwendet

Erweiterungen in CSS 2

- ◆ **Es lässt sich ermitteln, welche Elemente benachbart sind**
`abb + absatz {text-indent: 2cm }`
nur der nach einer Abbildung folgende Absatz soll um 2 cm eingerückt werden
 - ◆ **Abfragen der Attributwerte**
`absatz[align="left"] {text-indent: 1cm }`
nur der Absatz mit dem Attribut „left“ wird entsprechend eingerückt
`absatz[test="no"] {display: none }`
alle Absätze mit dem Attribut test=„no“ werden nicht dargestellt
 - ◆ **über ein Stylesheet lässt sich (über `content:`) Text hinzufügen**
`absatz:after {`
 `display: block;`
 `content: "Absatzende";}`
hinter jedem Absatz wird der Text „Absatzende“ eingefügt
`kapitel para ueberschrift:before {`
 `content: counter(kapzaehler) ". ";`
 `counter-increment: kapzaehler;}`
vor jeder Kapitelüberschrift wird eine Kapitel-Nummerierung eingefügt
-

Erweiterung in Opera

- ◆ Die CSS 2 Funktionalität bietet derzeit nur der **Opera 5.12**
 - ◆ **Präsentationsmodus in Opera 5**
`@media projection { }`
alle hier vorgenommenen Einstellungen gelten nur für den Fall, dass in Opera der Vollbildmodus (F11) aktiviert wird
 - ◆ **Unterstützung von XML** derzeit nur über CSS, da offenbar ein XSLT-Prozessor den Browser zu groß werden lässt
-

Processing Instructions (PI)

- ◆ **PI** (Processing Instructions) erlauben es, Anweisungen für die Verarbeitung in einem XML-Dokument unterzubringen
 - ◆ Sie stehen zwischen den Zeichen `<? ... ?>` z. B.
`<?xml version="1.0" encoding="ISO-8859-1"?>`
 - ◆ **xml** wird hier als Schlüsselwort festgelegt und muss fortan stets am Anfang der PIs stehen
 - ◆ die PIs stellen hier Verarbeitungshinweise dar, in denen festgelegt wird, welche Stylesheets zum Einsatz kommen
`<?xml-stylesheet type="text/css2" href="t1.css"?>`
 - Sie entspricht der HTML-Anweisung
`<LINK href="mystyle.css" rel="stylesheet" type="text/css">`
 - ◆ daraufhin schickt der Web-Server die CSS-Datei mit an den Client
 - ◆ Die **PI** `xml-stylesheet` muss im Prolog der XML-Datei stehen
 - ◆ Es sind durchaus mehrere derartige PIs erlaubt
-

Namensräume (*Namespaces*)

- ◆ **Namensräume dienen als eine Möglichkeit**
 - Für die Hersteller, um die Zusammenstellung von Dokumentteilen zu bewerkstelligen
 - Für die *Working Groups*, um Namen zu reservieren, ohne die Freiheit des Anwenders in seiner Tag-Wahl zu beeinträchtigen
 - ◆ **Namensräume werden gekennzeichnet durch zwei Namen, die durch eine Kolon getrennt werden**
`xmlns:html`
 - ◆ **Der Namespace-Prefix wird als eine Abkürzung für den nachfolgenden URI angesehen**
`<Slide xmlns:html="http://www.w3.org/TR/REC-html40">`
Hier ist `html` als Entsprechung für den URI der HTML-Beschreibung zu sehen; so dass innerhalb des `Slide`-Elementes stets `html` der Namensraum ist, dessen Verantwortlichkeit durch den URI beschrieben wird
 - ◆ **Die Validierbarkeit wird davon nicht berührt**
 - ◆ **Einsatz von Namensräumen sind nur in XML-Instanzen, nicht aber in DTD's vorgesehen**
-

XML-Link

- ◆ Generell besteht auch in XML die Möglichkeit, Daten aus anderen Dokumenten in das eigene XML-Dokument einzublenden
 - ◆ Als eine einfache Rekonstruktion des `<a>`-Elementes in HTML
 - ◆ **XML-Link** spezifiziert eine zweiseitige Verknüpfung
`<Button xml:link="simple" href="slide2.xml">`
In diesem Fall wird ein anklickbarer Bereich `Button` festgelegt
 - ◆ Folgendes bietet **XML-Link**:
 - Beschreibt Links mit einem Link-Element
 - Identifiziert Links und deren Ziele durch ihren Typ und ihre Rolle
 - Lokalisiert Links durch eine mächtige Syntax
 - Legt das Standard-Verhalten fest
-

XML und CSS – Beispiel 1

```
<?xml version="1.0"?>
<?xml-stylesheet href="slides.css" type="text/css"?>
<!DOCTYPE Slide>
<Slide xmlns:html="http://www.w3.org/TR/REC-html40">
<html:img src="fhdw.gif" width="61" height="131" id="logo" />
  <SlideBody>
    <Heading>XML</Heading>
    <SubHeading>Grundlagen - Unterscheidung zu HTML</SubHeading>
    <List>
      <ListItem>Unterscheidung zwischen Form und Inhalt</ListItem>
      <ListItem>prozedurales vs. generisches Markup</ListItem>
      <ListItem>Uebliche Visualisierungsprobleme</ListItem>
      <ListItem>das laeuft ja schon hervorragend mit XML und
Folien</ListItem>
    </List>
  </SlideBody>
  <Bar>
    <Button xml:link="simple" href="next.gif">naechste Folie</Button>
  </Bar>
</Slide>
```

XML und CSS – Beispiel 1

```
slide { display: block }

slideBody {
  display: block;
  padding-left: 4px;
  border-left: 80px solid rgb(0,128,200);
  border-top: 4px solid rgb(0,128,200);
  border-bottom: 4px solid rgb(0,128,200);
  font-family: Verdana;
  font-size: 16pt;
  width: 940px;
  height: 618px;
  margin: 0px;
}
Heading {
  text-align: center;
  text-decoration: underline;
  font-size: 24pt;
  font-weight: bold;
  display: block;
  margin-top: 1em;
  margin-bottom: 0.75em;
}
```

XML und CSS – Beispiel 2

```
<?xml version="1.0"?>
<?xml-stylesheet href="email.css" type="text/css"?>
<mail>
  <Recipient>Marta Microbe</Recipient>
  <Sender>Arno Amoebe</Sender><br />
  <Date>Mon, 31. Jan 2000 010:30:00</Date>
  <Subject>XML-Kurs</Subject>
<Textbody>
  <p>Hallo <Name>Marta</Name>,</p>
  <p>Bitte schaue dir doch einmal den Aufsatz <br />
  "SGML, Java and the Future of the Web"<br />
  von <Name>Jon Bosak</Name> an.</p>
  <p>bis dann,</p>
  <p><Name>Arno</Name></p>
</Textbody>
</mail>
```

XML und CSS – Beispiel 2

```
mail      { display: block; background-color: rgb(255,255,255);
           margin-left: 10px; width: 500px }

Recipient { display: inline; font-weight: bold; color: green }
Sender    { display: inline; font-weight: bold; color: red;
           margin-left: 100px }

Date      { display: block; font-family: monospace;
           fontweight: bold; font-size: 10pt; margin-top: 15px;
           margin-bottom: 15px }

Subject   { display: block; font-weight: bold;
           font-family: helvetica; margin-left: 10px;
           margin-top: 5px; margin-bottom: 10px; width: 400px;
           border-style: solid; border-color: blue;
           text-align: center }

Textbody  { display: block; background-color: cyan;
           margin-left: 10px; width: 400px }

br        { display: block }
p         { display: block }

name      { font-style: italic; font-weight: bold }
```

XSL

- ◆ XSL steht für **Extensible Stylesheet Language** und ist eine auf XML basierende Sprache für Stilvorlagen
 - ◆ Von **Microsoft**, **Inso** und **Arbortext** initiiert und dem W3C vorgelegt
 - ◆ Ein Stylesheet dient dazu, einem Prozessor (Browser, etc.) Anweisungen zu geben, damit er die logische Struktur in eine Präsentationsstruktur überführen kann
 - ◆ XSL stellt Programmroutinen zur Verfügung, mit denen sich vom Ausgangsdokument abweichende Strukturen aufbauen lassen
 - ◆ Erste Realisation war der Prozessor **msxsl** von Microsoft
 - ◆ Letztes **Draft** Mitte Januar 2000
-

XSL-Prozessoren

- ◆ **msxml** von Microsoft
 - Empfehlenswert ist der Einsatz ab der Version 3.0
 - ◆ **xalan** der Apache-Group
 - Wird im Apache-Projekt Cocoon eingesetzt
 - ◆ **Saxon** von Michael Kay
 - <http://saxon.sourceforge.net/>
 - ◆ **XT** von James Clark
 - Verfügbar entweder als Java-Klassen oder als Windows-Executable
 - ◆ **xsltproc** schneller in C geschriebener Prozessor unter Linux und Win32, ursprünglich für das Gnome-Projekt entwickelt
 - ◆ **FOP** von James Tauber bzw. der Apache-Group
 - Ziel ist die Formatierung nach PDF 1.3 (**P**ortable **D**ocument **F**ormat), MIF (**F**ramemaker) und SVG (**S**calable **V**ector **G**raphics)
 - Verfügbar in Form von Java-Klassen, benötigt **xalan** und **xerces**
-

MSXML-Versionen

- ◆ **Installation:** `msxml3_de.exe` (für die Version 3.0SP1) oder `msxml4.exe` (für Version 4)
 - ◆ **Installation der Datei `xmllnst.exe` in `windows\system` (win9x) bzw. `winnt\system32` (NT, W2k)**
 - ◆ **Aufruf von `xmllnst` registriert die neue DLL**
 - ◆ **Mit `xmllnst /u msxml4` wird die Registrierung wieder auf die ältere Version zurückgenommen**
 - ◆ **Versionen:**
 - **V. 2.0 – [...] – Internet Explorer 5.0, Office 2000**
 - **V. 2.5 SP1 – [8.0.5226] - Internet Explorer 5.5, Windows 2000**
 - **V. 3.0 – [8.0.7820.0] - Internet Explorer 6.0**
 - **3.0SP1 – [8.20.8730.1]**
 - **4.0 – [4.0.8411.1] - Technology Preview July Version**
 - ◆ **Es empfiehlt sich für XSLT-Transformationen generell der Einsatz ab Version 3.0**
 - ◆ **`msxsl.exe` ist die Kommandozeilen-Version für den Microsoft XSL-Prozessor (nur `msxml 3.0` und `4.0`)**
-

Aufruf einiger XSLT-Prozessoren

- ◆ Voraussetzung für die meisten XSLT-Prozessoren ist eine JVM möglichst ab 1.2

- ◆ **Xt:**

```
xt test.xml test.xsl -o test.html
```

- ◆ **Xalan:**

```
java -cp xalan-2.1.0.jar;xerces.jar  
org.apache.xalan.xslt.Process -in test.xml -xsl test.xsl  
-out test.html
```

- ◆ **Msxsl – Kommandozeilenversion:**

```
msxsl test.xml test.xsl -o test.html
```

- ◆ **Fop – Formating Object Processor:**

```
java -cp fop-0.20.1.jar;batik.jar;xalan-2.1.0.jar;  
xerces.jar;jimi-1.0.jar org.apache.fop.apps.Fop  
-fo test.fo -xml test.xml -xsl test.xsl [-awt | -pdf | -mif]  
test.pdf/.mif
```

bei der Option „awt“ wird eine Bildschirmdarstellung versucht

- ◆ bei JVM 1.1.8 müssen die JAR-Dateien im separaten Klassenpfad gesetzt werden
-

XSL/XSLT (1)

- ◆ **XSL besteht aus zwei Teilen**
 - **Eine Sprache zur Beschreibung von Transformationen von XML-Dokumenten (XSLT)**
 - **Eine Sprache zur Spezifizierung der Formatierungssyntax**
 - ◆ **Ein XSL-Prozessor benötigt als Eingabe eine XML-Datei und eine Stil-Anweisung und erzeugt daraus die gewünschte Präsentation in zwei Phasen:**
 - **Baumtransformation – die XML-Struktur des Ausgangsdokumentes (Baumstruktur) wird in eine andere überführt, den Ergebnisbaum**
 - **Die eigentliche Formatierung**
-

XSL/XSLT (2)

- ◆ die Baumtransformation erfolgt durch Angabe von Regeln
 - ◆ Jede Regel hat einen **Pattern**-Teil, der die möglichen Anwendungsstellen im XML-Strukturbaum festlegt und einen **Template**-Teil, der bestimmt, durch welche Struktur die Anwendungsstelle im Ergebnisbaum ersetzt werden soll
 - ◆ Der Transformationsprozess ermöglicht den flexiblen Zugriff auf bestimmte Bereiche des XML-Dokuments.
 - Damit lassen sich aus einer XML-Instanz durch entsprechende XSL-Anweisungen Dokumente für jeweils unterschiedliche Ausgabegeräte erstellen
 - → ideal für die Strukturierung umfangreicher Web-Angebote
s. **Cocoon**-Projekt
-

XSLT

- ◆ Üblicherweise erfolgt die Transformation in eine Struktur, die bereits darstellungsorientiert ist und möglichst auf einer XML-Struktur aufbaut, d.h. der Ergebnisbaum ist ebenfalls XML
 - XML → HTML
nicht so ideal, da HTML auf SGML aufbaut
 - XML → XHTML, HTML auf Basis von XML
 - XML → WML (**W**ireless **M**arkup **L**anguage), eine in XML verfasste Sprache für Dokumente, die von sog. WAP-Handys dargestellt werden können
 - ◆ Soll eine geräteunabhängige Formatierung für Print/Online-Medien erzielt werden, benötigt man zusätzlich den Formatierungsprozess durch **XSL:FO**
 - ◆ FO steht für **F**ormating **O**bjects
 - ◆ Formatierungsobjekte sind Rechtecke, die bestimmte Eigenschaften aufweisen
-

XSL vs. CSS

- ◆ **XSL baut auf XML-Syntax auf**
 - ◆ **Ziel ist eine möglichst einfache Nutzung im Internet (daher nicht DSSSL)**
 - ◆ **CSS bietet nur Formatanweisungen für die entsprechenden Tags**
 - ◆ **In XSL muss der zu formatierende Strukturbaum nicht gleich dem Ergebnisbaum sein, es bietet somit die Möglichkeit zu weitreichenden programmtechnischen Manipulationen**
 - ◆ **Beide Verfahren bieten einen Mechanismus, die jeweiligen XML-Elemente zu präsentieren**
 - **CSS: über Selektoren und Eigenschaften**
 - **XSL: über Pattern und Formatting Objects**
 - ◆ **Verfügbare Programme:**
 - **XSL wird derzeit nur vom IE und serverseitigen Publishing Systemen unterstützt**
 - **CSS hingegen von fast allen modernen Browsern (IE, Mozilla, Opera) und Editor-/Textsystemen, die XML unterstützen**
-

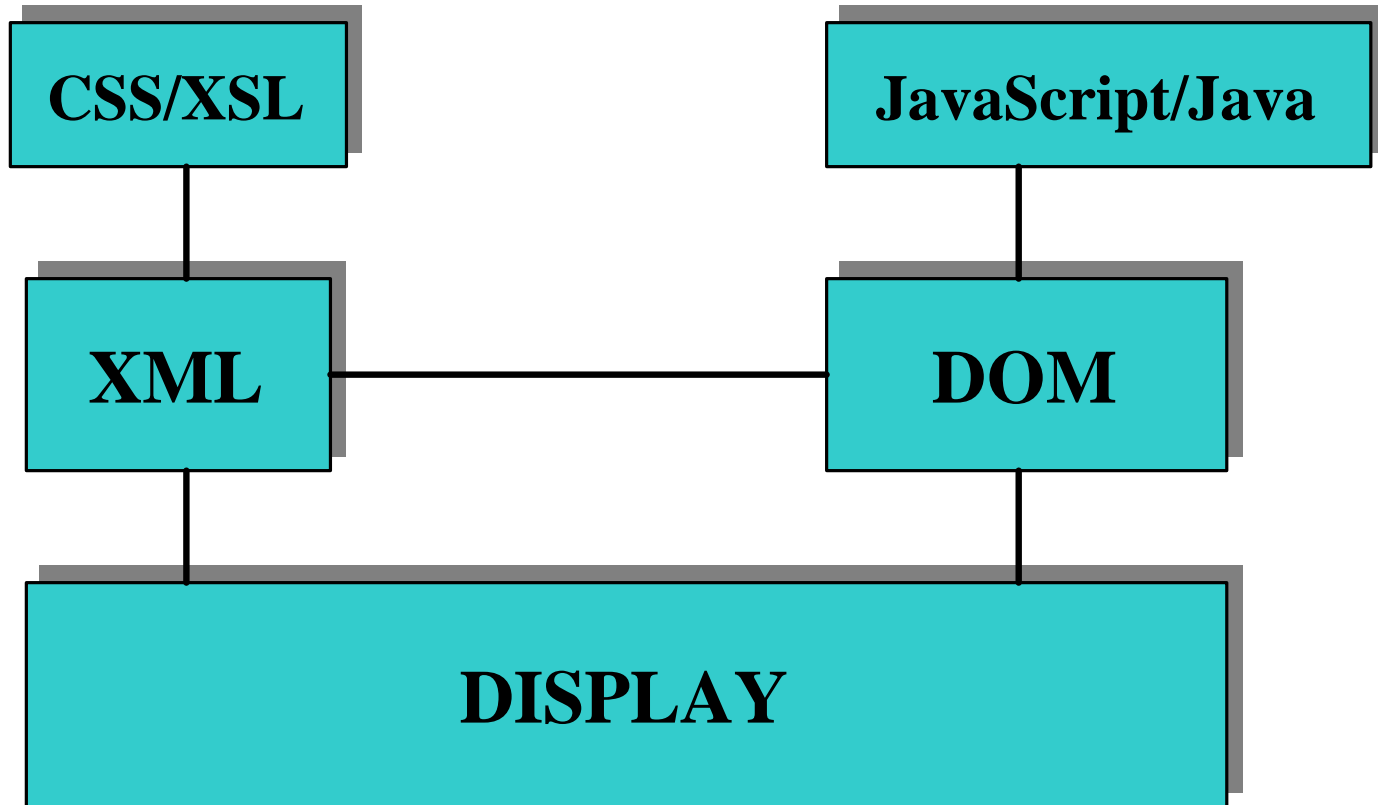
DSSSL

- ◆ DSSSL steht für **D**ocument **S**tyle **S**emantic and **S**pecification **L**anguage
 - Internationaler Standard ISO/IEC 10179:1996
 - ◆ Ein an dem Scheme-Lisp-Dialekt angelehnte Programmiersprache
 - ◆ Eine mächtige und daher schwer zu erlernende Sprache
 - ◆ Besteht aus mehreren Komponenten:
 - **Style Language** zur Formatierung von SGML/XML-Dokumenten
 - **Transformation Language** zur Umwandlung von SGML/XML-Dokumente, die einer DTD entsprechen in solche, die einer anderen entsprechen
 - **Query Language** zur Möglichkeit der Abfrage von Elementen und Attributen
 - ◆ Ermöglicht somit auch die Transformation von XML nach HTML
 - ◆ Als Tool kommt hierfür zum Einsatz James Clarks Programm **Jade** (von James' **DSSSL Engine**)
<http://www.jclark.com/jade/>
-

DOM

- ◆ **DOM** steht für **D**ocument **O**bject **M**odel und beschreibt in einer objektorientierten Annäherung eine Web-Seite als eine Ansammlung von Objekten, die einzeln adressierbar sind
 - Die Web-Seite kann hierbei als HTML- aber auch als XML-Objekt begriffen werden
 - Die Modularisierung in Objekten ermöglicht deren Wiederverwendbarkeit
 - Objekttypen als Vorlage (*template*) und die Instanzen als die jeweilige konkreten Realisierungen
 - ◆ Für den Zugriff auf diese Objekte eignen sich bestens objektorientierte Programmiersprachen wie **Java**, **Python**, aber auch **JavaScript**, **TCL**, **Perl**
 - ◆ **API** (**A**pplication **P**rogramming **I**nterface) als eine Sammlung von Regeln, wie die einzelnen Objekte erreicht und verändert werden können
 - ◆ Hieraus ergeben sich im wesentlichen drei Funktionen des DOM:
 - **WER**: wie werden XML-Objekte hier repräsentiert
 - **WAS**: was können die Objekte und wie hängen sie miteinander zusammen
 - **WIE**: wie können diese Objekte adressiert werden
 - ◆ Während XML die Daten strukturiert und identifiziert, ermöglicht das DOM die Manipulation über Programme, d. h. die Aktion der Daten untereinander
-

Zusammenspiel von **XML** und **DOM** für eine dynamische Web-Präsentation



XHTML

- ◆ **XHTML: EXtensible HyperText Markup Language**
 - **August '99 Proposed Recommendations for XHTML 1.0 Working Draft am 24.12.1999**
Neuformulierung am 24.1.2000
 - ◆ **Während HTML als Applikation von SGML entwickelt wurde, ist XHTML eine Applikation von XML**
 - ◆ **Gründe für XHTML:**
 - **Verbreitete Nachlässigkeit bei der Verwendung von HTML-Tags**
 - **Unvollkommene Implementierung der Stilvorlagen**
 - **Neue Browser-Arten erfordern modifizierte/reduzierte HTML-Sprache**
 - **Bessere Integration von HTML in andere Tag-orientierte Sprachen**
 - ◆ **XHTML erfordert Wohlgeformtheit und ist somit Parser-gerecht**
 - ◆ **Die Tags und Attribute müssen kleingeschrieben werden!**
-

Tidy

- ◆ Programm von Dave Raggett zur Umwandlung von HTML-Dokumenten nach XHTML
<http://www.w3.org/People/Raggett/tidy>
 - ◆ Korrektur der HTML-Auszeichnung in Richtung Wohlgeformtheit
 - ◆ Unbekannte Sprachelemente und Attribute werden aufgezeigt
 - ◆ Der Quellcode wird entsprechend eingerückt ausgegeben (pretty printing)
 - ◆ Darstellung der Sonderzeichen: US ASCII, ISO Latin-1, UTF-8 in HTML-Entitäten
 - ◆ Presentational Markups werden möglichst in CSS-Notation überführt
 - ◆ Erstellung von Slides, indem bei jeder <h2>-Umgebung eine neue Datei erzeugt wird
 - ◆ Binaries für Win32, Mac, RPMs für Linux, Quellen für C und Java
Plugin für den freien HTML-Editor [Phase 5²](#)
`tidy -f err.txt -im foo.html`
korrigiert die Datei *foo.html*, rückt die Sprachelemente ein und gibt etwaige Fehler in *err.txt* aus
-

Tidy - Optionen

-config <i>file</i>	liest die Konfigurationsdatei <i>file</i> ein
-indent <i>oder</i> -i	Einrückung der Sprachelemente
-omit <i>oder</i> -o	läßt optionale Ende-Tags weg
-wrap 72	Zeilenumbruch nach 72 (Standard ist 68)
-upper <i>oder</i> -u	Großschreibung der Tags (Standard ist klein)
-clean <i>oder</i> -c	ersetzt font, nobr & center durch CSS
-raw	keine Entitäten für ASCII 128 - 255
-ascii	Ausgabe ASCII, Eingabe Latin-1
-latin1	Aus-/Eingabe Latin-1
-utf8	Aus-/Eingabe UTF-8
-numeric <i>oder</i> -n	Ausgabe von numerischen Entitäten
-modify <i>oder</i> -m	Veränderung an der Originaldatei
-errors <i>oder</i> -e	nur Ausgabe der Fehlermeldungen
-f <i>file</i>	gibt Fehler in <i>file</i> aus
-xml	Eingabedatei ist XML
-asxml	konvertiert HTML nach XML
-slides	Folienausgabe getrennt durch das <h2>-Element
-help	Kommandooptionen

Publishing Framework

- ◆ mittelfristig wird jede größere Applikation Web-fähig sein müssen
 - ◆ Problem 1:
 - Die Konstruktion von aufwendigen Web-Interfaces zieht zeitraubende HTML-Programmierung nach sich
 - Dies kompliziert sich noch durch die Verfügbarkeit unterschiedlicher Devices (Handys, Print-Medien, etc.)
 - ◆ Lösung: ein Publishing-System, das die Dokumente in XML vorhält und die Anpassung an die Ausgabegeräte über XSL/XSLT ermöglicht
 - ◆ Problem 2: mittels **JSP** (Java Server Pages) lässt sich zwar elegant Java-Code in HTML-Seiten einbetten, hier wird aber nicht zwischen Inhalt und Darstellung getrennt und weiterhin auch nicht die Transformation in andere Formate ermöglicht
 - ◆ Lösung: **XSP** (Extensible Server Pages) baut auf XML auf, erlaubt somit die Trennung des Inhaltes von der Darstellung und weiterhin lassen sich die XSP-Prozesse durch Stylesheets in ein anderes Format transformieren
-

Cocoon (1)



- ◆ **Apache Cocoon-Projekt, begründet von Stefano Mazzocchi**
<http://xml.apache.org/cocoon>
 - ◆ **basiert vollständig auf Java, als Java-Servlet lässt es sich in jede Servlet-Engine eines Web-Servers integrieren (z. B. im Apache mit **Apache-Jserv**)**
 - ◆ **In Abhängigkeit vom Ausgabegerät werden die unterschiedlichen Formate zur Laufzeit generiert und ausgeliefert**
 - ◆ **Funktion:**
 - **Der XML-Parser **xerces**, überführt ein vorliegendes XML-Dokument in eine interne Baumstruktur, der **DOM**-Repräsentation**
 - **Diese Datenstruktur transformiert der XSL-Prozessor **xalan** und erzeugt eine Ausgabe**
 - **Zur Print-Ausgabe dient der Formatierer **FOP (Formatting Object Processor)****
 - **Für komplexere Dokumente und als Bindeglied zu Cocoon 2 dient das Apache **Stylebook****
-

Cocoon (2)

- ◆ Aktuelle Version 1.81,
Version 2 befindet sich noch in der Entwicklung
- ◆ Orientiert sich vollständig an den W3C-Standards
- ◆ Liefert verschiedene Ausgabeformate: derzeit **HTML**, **WML**, **PDF**, **SVG**
- ◆ Unterstützt unter anderem folgende Ausgabeprogramme/-geräte:
MS Internet Explorer, Opera, Lynx, Nokia (WAP), UP (WAP),
Netscape/Mozilla
- ◆ Macht den Web-Server XML-fähig
Ermöglicht den Weg hin zu eine bessere
Strukturierung der Web-Site und fördert die
Arbeitsteilung, um die drei unterschiedlichen
Komponenten (Inhalt, Struktur, Format)
zu erstellen

