XML
Extensible Markup Language

Was ist XML?

- eine „erweiterbare Auszeichnungssprache“: extensible markup language
- eine Metasprache, also eine Sprache zur Beschreibung von Sprachen
- HTML minus Präsentation plus eigene Tags
- ein halbstrukturiertes Datenmodell
  - oder eher nur eine selbsterklärende Austauschsyntax?
- das ASCII des Webs
- viele gute (und ein paar unnütze) Informatiker-Ideen

XML

- Ursprung: HTML und SGML (ISO-Standard, 1986, ~600 Seiten)
- W3C-Standard (~26 Seiten): XML-Syntax + DTDs*
  http://www.w3.org/TR/REC-xml
- XML bietet eine **Teilmenge** des SGML-Funktionsumfangs
- HTML ist eine **Anwendung** von SGML (also eine aus SGML heraus produzierte Sprache)
- XHTML ist eine Anwendung von XML

* DTD = Document Type Definition

Datenformate

ÐÏ^Qà¡±^Zá^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@>^@^C^@þÿ
^@^F^@^@^@^@^@^@^@^@^@^@^A^@^@^@#^@^@^@^@^@^@^@
^@^P^@^@%^@^@^@^A^@^@^@þÿÿÿ^@^@^@^@"^@^@^@ÿÿÿÿÿÿÿ
ÿÿ
ÿÿ
ÿÿ
ÿÿ
ÿÿÿì¥Á^@q^@^D
ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ^@^@^@^R¿^@^@^@^@^@^@^P^@^@^@^@^@^D^@^@
Ç^G^@^@^N^@bjbjt+t+^@^@^@
^@Some Quotations from the Universal Library^M1 Famous Quotes^M1.1 By William I^M[2, Sonnet
XVIII]^MShall I compare thee to a summer's day?^MThou art more lovely and more
temperate.^MRough winds do shake the darling buds of May,^MAnd summer's lease hath all too
short a date.^MSometime too hot the eye of heaven shines,^MAnd often is his gold complexion
dimmed.^MAnd every fair from fair some declines,^MBy chance or nature's changing course
untrimmed.^MBut thy eternal summer shall not fade,^MNor lose possession of that fair thou
owest,^MNor shall Death brag thou wander'st in his shade^MWhile in eternal lines to time thou
growest.^MSo long as men can breathe, or eyes can see,^MSo long live this, and this gives
life to thee.^M1.2 By William II^M[1, p.265]^M\223The obvious mathematical breakthrough
would be development of^Man easy way to factor large prime numbers
."^MReferences^M[1] W. H. Gates. The Road Ahead. Viking Penguin, 1995.^M[2] W.
Shakespeare. The Sonnets of Shakespeare.609.^M^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^@^@^@^
ÿÿÿ^A^@þÿ^C^@^@ÿÿÿÿ^F^B^@^@
^@^@^@À^@^@^@^@^@^@F^X^@^@^@Microsoft Word Document^@^@^@^@
MSWordDoc^@^P^@^@^@Word.Document.8^@ô9²q^@^@^@^@^@^@^@^@
^@
^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^

Datenformate

```
\documentclass{article}
 \begin{document}
 \title{Some Quotations from the Universal Library}
...
\section{Famous Quotes}
 \subsection{By William I}
 \textbf{\cite[Sonnet XVIII]{shakespeare-sonnets-1609}}
    \begin{verse}
     Shall I compare thee to a summer's day?\\
     Thou art more lovely and more temperate. \\
     Rough winds do shake the darling buds of May, \\
     And summer's lease hath all too short a date. \\
     Sometime too hot the eye of heaven shines, \\
     And often is his gold complexion dimmed. \\
   …
     \qquad So long as men can breathe, or eyes can see,\\
     \qquad So long live this, and this gives life to thee.   \\
   \end{verse}
...
 \bibliographystyle{abbrv}
...
\end{document}
```

Datenformate

```
<HTML>
<HEAD>
<TITLE>Some Quotations from the Universal Library</TITLE>
</HEAD>

<BODY>

<B><FONT FACE="Arial" SIZE=5><P>Some Quotations from the Universal Library</P>
</FONT><I><FONT FACE="Arial"><P>1 Famous Quotes</P>
</B></I><P>1.1 By William I</P>
<B><P>[2, Sonnet XVIII]</P></B>
<P>Shall I compare thee to a summer's day?</P>
<P>Thou art more lovely and more temperate.</P>
<P>Rough winds do shake the darling buds of May,</P>
<P>And summer's lease hath all too short a date.</P>
<P>Sometime too hot the eye of heaven shines,</P>
<P>And often is his gold complexion dimmed.</P>
...
</BODY>
</HTML>
```

Datenformate

```
<?xml version="1.0"?>
<universal_library>
  <books>
    <book><title>Some Quotations from the Universal Library</title>
      <section><title>Famous Quotes</title>
        <subsection><title>By William I</title>
          <quote bibref="shakespeare-sonnets-1609">
          <title>Sonnet XVIII</title>
          <verse>
            <line>Shall I compare thee to a summer's day?</line>
            <line>Thou art more lovely and more temperate. </line>
            <line>Rough winds do shake the darling buds of May, </line>
          </verse>
  …
      <subsection> <title>By William II</title>
        <quote bibref="gates-road-ahead-1995">
          <title>Page 265</title>
          <line>``The obvious mathematical breakthrough would be development of an easy way to factor large prime numbers.''</line>
        </quote>
      </subsection>
      </section>
</book>  …
</books>
</universal_library>
```

HTML gegen XML

```
<h1>Bibliography</h1>
<p><i>Foundations of DBs</i>, Abiteboul, Hull, Vianu
<br>Addison-Wesley, 1995
<p><i>Logics for DBs and ISs</i>, Chomicki, Saake, eds.
<br>Kluwer, 1998
```

HTML
fest vorgegebene Menge von Marken (Tags); beschreiben Aussehen und grobe Dokumentstruktur

Definiert, wie der Text aussieht

```
<bibliography>
   <book><title>Foundations of DBs</title>
      <author>Abiteboul</author>
      <author>Hull</author>
      <author>Vianu</author>
      <publisher>Addison-Wesley</publisher>
       ...
    </book>
    <book>   ... <editor>Chomicki</editor>... </book>     ...
</bibliography>
```

XML
frei wählbare Menge von Marken, semantische Angaben

Definiert, woraus der Text besteht

Selbsterklärendes Datenformat

- XML kann leicht geparst werden
  - beinhaltet seinen eigenen Parsebaum in den Daten
  - Aufbau (und Semantik) kann häufig leicht rekonstruiert werden
- Mit Hilfe weiterer Metasprachen (z.B. DTD) können gültige Ausdrücke definiert werden
- XML trennt Präsentation von Inhalt
  - Präsentation über Stylesheets
  - Möglichkeit, verschiedene Stylesheets anzuwenden
    - HTML
    - PDF ...

Perspektiven

- Die Rolle von XML wird unterschiedlich wahrgenommen
  - Dokumente
    - Daten = lineare Textdokumente
    - XML dient zum Auszeichnen von Textteilen und bringt Semantik und Struktur in den Text
  - Datenbanken
    - XML ist das prominenteste Beispiel eines halbstrukturierten Datenmodells (d.h. das Datenmodell steckt in den Daten)
    - XML deckt das Spektrum zwischen unstrukturierten, regulären und voll strukturierten Daten ab.

XML-Begriffswelt

- XML (Extensible Markup Language)
- Namensräume (XML Namespaces)
- XML-DTDs (Document Type Definition)
- RDF (Resource Description Framework)
- XSL (Extensible Stylesheet Language)
- XPath, XPointer, XLink
- XQL, XML-QL (XML Query Language), Quilt
- XMAS (XML Matching And Structuring language)
- eXcelon, ...

Hilfsprogramme und Standards

XML-Anwendungen

- Fach- und branchenspezifisch (vertikal)

Werbung	adXML	Online-Werbung
Literatur	Gutenberg	Übertragung literarischer Werke
Verzeichnisse	dirXML	Novells Directory Services Markup Language
Webserver	apacheXML	Parser, XSL, Veröffentlichen
Reisen	openTravel	Informationen zu Fluglinien, Hotels, Autovermietern
Nachrichten	NewsML	Nachrichten formatieren und liefern
Personal	XML-HR	Standardisierung u.a. für Bewerbungsdaten
Entwicklung	IDML	Unterstützung nachhaltiger Entwicklung
Sprache	VoxML	Markup für sprachbasierte Anwendungen
Wetter	OMF	Wetterbeobachtung
Geodaten	ANZMETA	Geoinformationssysteme
Banken	MBA	Mortgage Bankers Association of America
Gesundheit	HL7	DTDs für Verschreibungen und andere Informationen
Mathematik	MathML	Mathematik-Markupsprache
Soziologie	DDI	Data Documentation Initiative

Rahmenwerke (Frameworks)

- Anwendungen im E-Commerce
  - eCo Framework: XML-Spezifikationen zur Unterstützung der Interoperabilität im E-Business
  - Commerce One: Common Business Library (CBL), eine Sammlung von Geschäftsprozesskomponenten (als DTD, XDR, SOX)
  - BizTalk: Rahmenwerk von Microsoft; erlaubt automaisierte Verarbeitung von Geschäftsdokumenten
  - cXML (Commerce XML) – Sammlung von Tags für E-Procurement in BizTalk
- Dient der Kommunikation innerhalb und zwischen Branchen

Datenaustausch

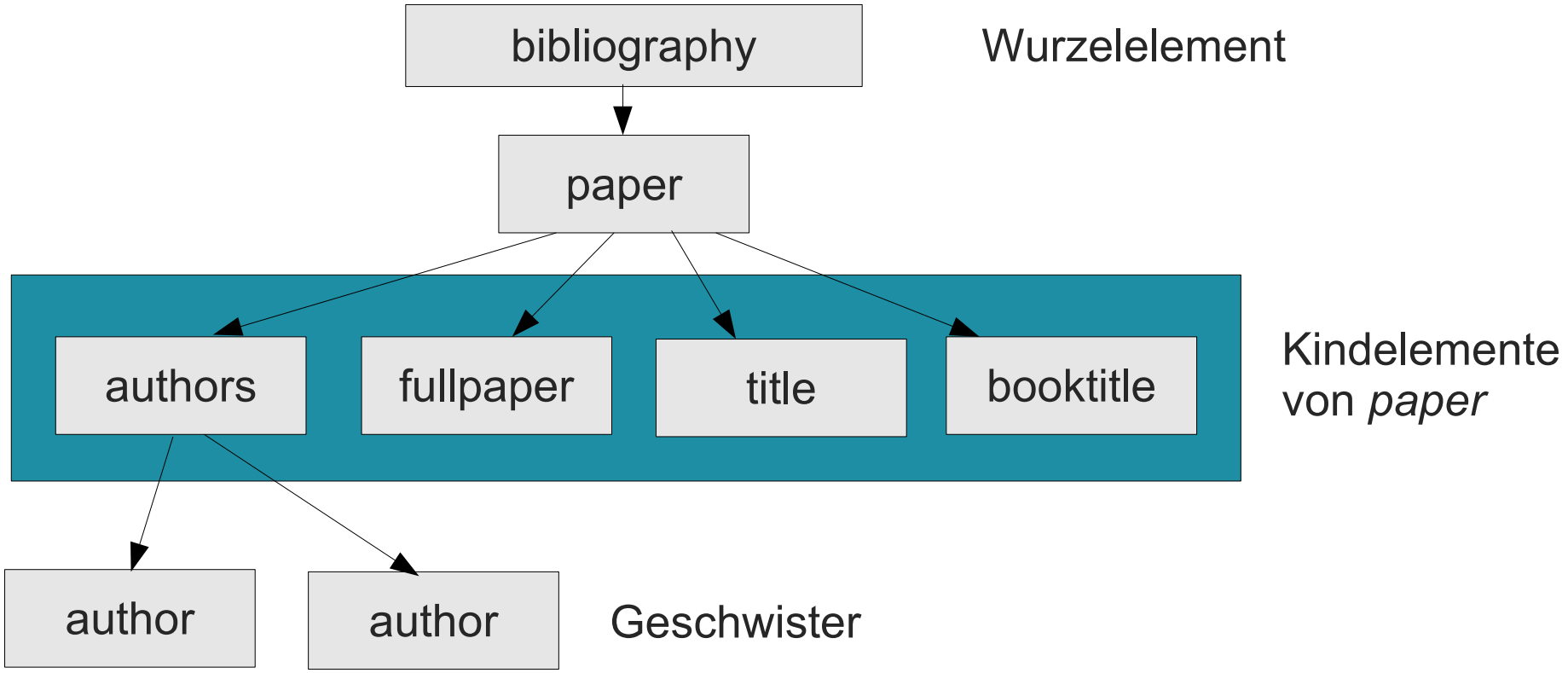
- Electronic Data Interchange (EDI)
  - RosettaNet: Einheitliches Format für Online-Bestellungen
  - FpML (Financial products Markup Language): Austausch von Finanzdaten
- Open Buying on the Internet (OBI)
  - OBI: Unterstützung von hochvolumigen Transaktionen im B2B-Bereich im Internet
- VISA Invoices
  - VISA-Invoice-Spezifikation bietet eine umfassende Liste von Datenelementen für die Rechnungsstellung

Struktur

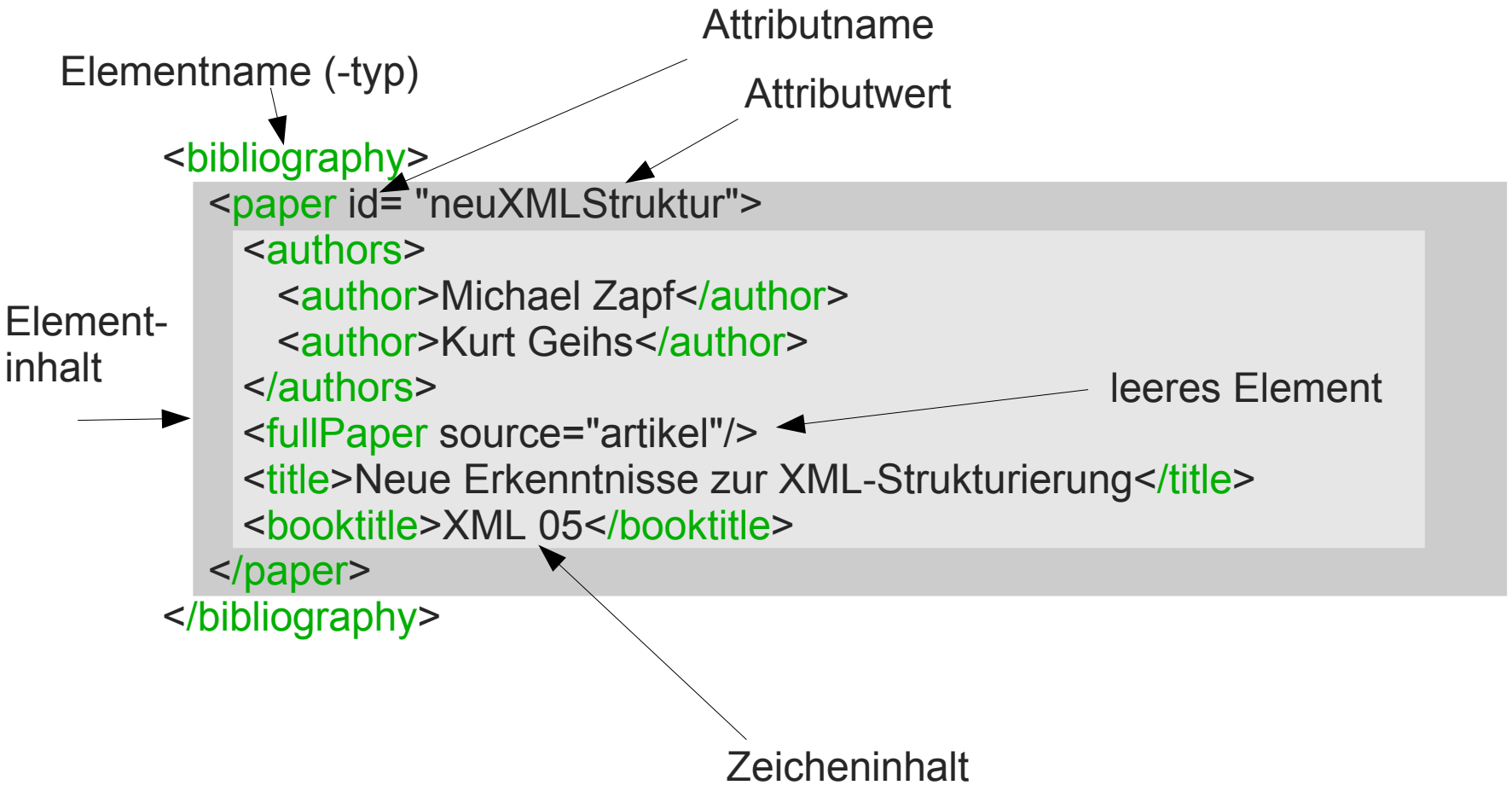
- Marken erlauben eine strukturierte Darstellung des Dokumentinhalts bereits als XML

```
<bibliography>
   <paper ID= "neueXMLStruktur">
     <authors>
       <author>Michael Zapf</author>
       <author>Kurt Geihs</author>
     </authors>
     <fullPaper source="artikel"/>
     <title>Neue Erkenntnisse zur XML-Strukturierung</title>
     <booktitle>XML 05</booktitle>
   </paper>
   <!-- Das ist ein Kommentar -->
</bibliography>
```

Logischer Aufbau



Elemente und Inhalt



Elemente und Attribute

- Zwei Grundkonzepte: Element und Attribut
- Element
  - Knoten im Baum
  - kann Teilbaum beinhalten (d.h. Element) oder leer sein
  - Kindknoten können mehrfach auftauchen
  - repräsentiert einen Bestandteil des Dokuments
- Attribut
  - zu einem Element gehörend
  - nur einmal pro Element vertreten
  - Inhalt nicht strukturiert, nur einfache Zeichenkette
  - repräsentiert ein Verhältnis zwischen einem Element und einer weiteren Information

```
<buch>
  <titel>XML</titel>
  <inhalt>...</inhalt>
</buch>
```

```
<buch titel="XML">
  <inhalt>...</inhalt>
</buch>
```

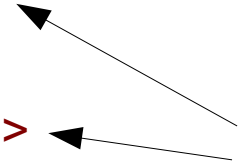
Namensräume

- Element- und Attributbezeichnungen können unterschiedliche Bedeutungen haben
  - <element> kann in der Chemie ein chemisches Element bezeichnen
  - <element> kann ein geometrisches Objekt bezeichnen
- XML-Namensräume schaffen Klarheit
  - definieren das verwendbare Vokabular
  - Format:
    <Namensraum:Elementname ...>
    <... Namensraum:Attributname ...>

Namensräume

- Mechanismus für global eindeutige Namen

```
<h:html xmlns:xdc="http://www.xml.com/books"
        xmlns:h="http://www.w3.org/1999/xhtml">
<h:head><h:title>Book Review</h:title></h:head>
   ...
<xdc:bookreview>
  <xdc:title>XML: A Primer</xdc:title>
  ...
</xdc:bookreview>
...
</h:html>
```



Verwechslung durch Namensräume vermieden

- Namensräume identifizieren lediglich das Vokabular
- Zusätzliche Mechanismen erforderlich, um Bedeutung und Struktur der Tags zu definieren

Wohlgeformte Dokumente

- Ein XML-Dokument nennt man „wohlgeformt“ (well-formed), wenn es
  - mit einer XML-Deklaration beginnt
    <?xml version=“1.0“?> (Momentan ist 1.0 maßgeblich; es gibt aber schon 1.1)
  - mindestens ein Datenelement aufweist
  - genau ein Datenelement alle anderen Datenelemente als Abkömmlinge beinhaltet (Wurzelelement, XML-spezifisch: **Dokumentelement**)
  - Elemente korrekt geschachtelt sind: **<a> ... ... **
  - alle Attribute in Anführungszeichen einschließt
  - für Sonderzeichen spezielle XML-Entitäten verwendet werden (z.B. „**<**“ für „<“)
- Genauer: Siehe Grammatik (http://www.w3.org/TR/REC-xml/)

Wohlgeformte Dokumente

- Wichtig:
  - Keine Vorgaben, wie die Elementnamen lauten (auch nicht für das Wurzelelement!)
  - Alle Elemente müssen geschlossen werden
    - bei leeren Elementen: „**<beispiel attr="xxx"/>**"
- Jedoch ist nicht klar, ob die Elemente sinnvoll zu einem Dokument zusammengestellt sind

```
<?xml version="1.0">
<zeit>
  <farbe gewicht="true">
    <hoehe>salzig</hoehe>
  </farbe>
  <farbe>1</farbe>
</zeit>
```

Welche Elemente sind erlaubt?
Welche Attribute sind erlaubt?
Welche Werte sind erlaubt?

→ **Dokumenttyp**

Dokumenttypen

- Erstellen von Dokumenttypdefinitionen (DTDs)

  Ein **wohlgeformtes** Dokument, das die Regeln einer DTD beachtet, wird als **gültiges** (valid) Dokument (bezüglich der DTD) bezeichnet

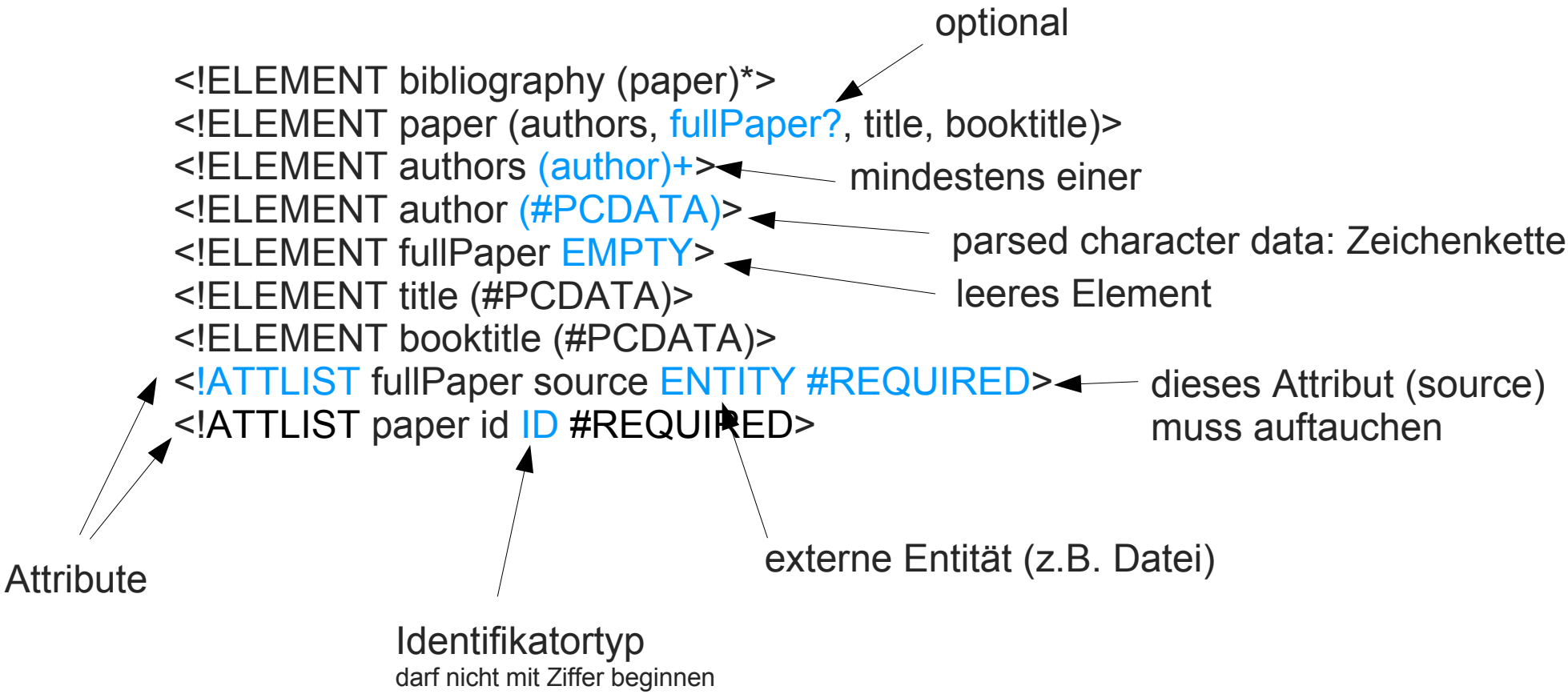
- DTDs kann man über kontextfreie Grammatiken beschreiben

document	→	<bibliography>paper*</bibliography>
paper	→	<paper id="text">authors [fullpaper] title booktitle</paper>
authors	→	<authors>author [author*]</authors>
author	→	<author>text</author>
fullpaper	→	<fullPaper source="path"/>
title	→	<title>text</title>
booktitle	→	<booktitle>text</booktitle>

text, path: Zeichenketten aus dem gültigen Zeichensatz

Dokumenttypen

- Daraus entsteht diese DTD:



DTD: Elemente

Anzahl:

(element)	1mal element
(element)?	0- oder 1mal element
(element)*	0- oder mehrmals element
(element)+	1- oder mehrmals element

Reihen und Alternativen:

(el1, el2, el3, ...)	Reihenfolge el1, el2, el3,...; jedes genau einmal und in dieser Reihenfolge
(el1 | el2 | el3 | ...)	Eines von el1 oder el2 oder el3 ...

Inhalt:

(#PCDATA)	Gewöhnliche Zeichenketten
EMPTY	Kein Inhalt
(#PCDATA | el1 | el2 | ...)*	Gemischter Inhalt aus Zeichenketten und den genannten Elementen
ANY	Alle Kombinationen erlaubt (Zeichen, Elemente, ...)

DTD: Attribute

Referenzen:

ID	Identifikator (darf nur einmal im Dokument auftauchen), muss mit Buchstaben anfangen!
IDREF(S)	nennt über ID definierte(n) Identifikator(en)

Datentypen:

ENTITY(/-IES)	externe Ressource(n) (Bild, Ton, Film, ...)
CDATA	Zeichenkette (enthaltene Zeichen bleiben unverändert, also auch Zeichen-/Entitätsreferenzen wie „#x20“ für Leerzeichen)
NMTOKEN(S)	Name(n) (muss nicht eindeutig sein)
NOTATION	Nicht-XML-Daten
(wort)	Zeichenkette „wort“ (nur diese)
(wort1 | wort2 | ...)	Eines von „wort1“, „wort2“ usw.; z.B. (true | false)

Verwendung:

#REQUIRED	Attribut notwendig
#IMPLIED	nicht notwendig
#FIXED	vorgegebener Wert (folgend)

Eines hiervon muss angegeben werden

Angefügter Wert ist Vorgabewert:

```
<!ATTLIST dokument name CDATA "unbenannt">
```

DTD-Einbindung

- Einbindung intern

```
<?xml version="1.0"?>
<!DOCTYPE bibliography [
        <!ELEMENT bibliography (paper)*>
        …
]>
<bibliography>…</bibliography>
```

Muss Wurzelelement sein

Dokumenttyp-definition

Dokumenttyp-deklaration

- oder extern

```
<?xml version="1.0"?>
<!DOCTYPE bibliography SYSTEM "bib.dtd">
<bibliography>…</bibliography>
```

wenn die DTD unmittelbar gefunden werden kann (Webadresse, Dateisystem), sonst PUBLIC (mit Identifikator), z.B.

```
<!DOCTYPE bibliography PUBLIC
Formal Public Identifier (RFC3151) → "-//Michael Zapf//DTD BIB v1.0//EN"
URI der DTD → "http://www.mizapf.de/standard.dtd">
```

DTD-Nachteile

- DTDs definieren die Struktur eines gültigen Dokuments
  - hilft bei der Erzeugung von Dokumenten, Abfragen
  - flexibel, halbstrukturiertes Datenmodell (Schachtelungen, ANY, Optionen, Wiederholungen,...)
  - aber: sehr Dokument-orientiert (Erbe von SGML)
  - keine Namensräume, Datentypen, Vererbung
  - keine Beschränkungen formulierbar (z.B. 3-5 Kindelemente)
- DTDs verfügen über eigene Syntax
  - noch aus SGML-Zeiten
  - kein wohlgeformtes XML-Dokument → nicht selbst validierbar

XML-Schema

- XML-Schema (W3C, Mai 2001)
  - orientiert sich an Datentypen
  - Auch XML Schema Definition (XSD) genannt
- Mächtiger als DTD
  - XML-Schemata weisen XML-Format auf; keine neue Syntax
  - Datentypen vorhanden: Standard und benutzerdefiniert
  - Vererbung
  - Nullwerte
  - Unterstützung für XML-Namensräume
  - Erweiterbar, unterstützt Modularisierung und Wiederverwendung

Datentypen

- Einfache Datentypen
  - 44 eingebaute Typen: date, int, float, string, ...
  - haben weder Attribute noch eingebettete Elemente (*atomische Typen*)
  - Wertebereich kann eingeschränkt werden
- Komplexe Datentypen
  - können mehrere Attribute oder Elemente beinhalten
  - Komposition (notwendig, wenn Elemente auftauchen)
    - all: alle enthaltenen Elemente genau einmal*
    - sequence: genau einmal* in dieser Reihenfolge
    - choice: genau einmal* eines der enthaltenen Elemente

    *Standardvorgaben
  - können innerhalb und außerhalb eines Elements definiert werden
    - referenzierbar über „name“

Beispiel

```
<?xml version="1.0"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="notiz">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="to" type="xs:string"/>
                <xs:element name="from" type="xs:string"/>
                <xs:element name="heading" type="xs:string"/>
                <xs:element name="body" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

komplexer, adhoc-definierter, unbenannter Datentyp (kommt also nur hier vor)

Element

- Definiert Format der Elemente
  - beinhaltet Elemente, die im XML-Dokument den enthaltenen Elementen entsprechen
  - Attribute:
    - name: Name des Elements im XML-Dokument
    - type: Datentyp des Elements (Wertebereich kann eingeschränkt werden)
    - default, final, minOccurs, maxOccurs, ref, ...
- Beispiele

```
<xs:element name=”geburtsdatum” type=”xs:date”/>
<xs:element name=”vorname” type=”xs:token” minOccurs=”1” maxOccurs=”unbounded”/>
<xs:element name=”pi” type=”xs:double” fixed=”3.141592” final=”#all”/>
```

token: bereinigter String (kein CR/LF/TAB, keine zwei Leerzeichen hintereinander, kein Leerzeichen am Anfang oder Ende)

Attribut

- Definiert das Format der einem Element zugehörenden Attribute
- Einige Attribute von <attribute>
  - name, type: Name des Attributs und entsprechender Datentyp
  - fixed: Fester Wert
  - ref, use, ...

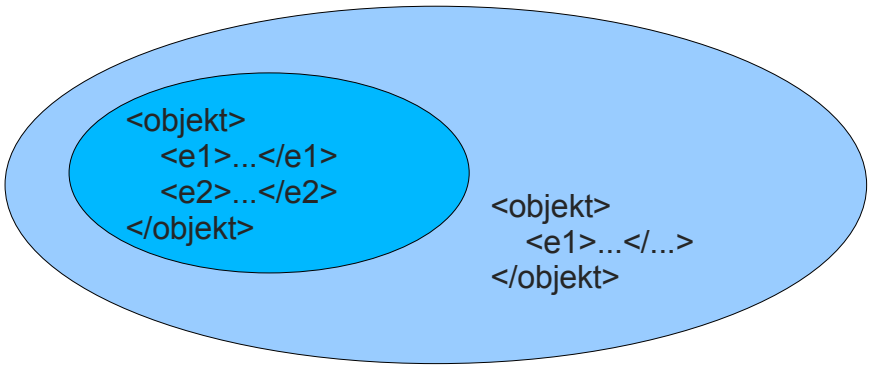
```
<xs:attribute name=”myAttribute1”/>
<xs:attribute name=”myAttribute2” type=”xs:decimal”/>
<xs:attribute name=”myAttribute3”>
        <xs:simpleType>
                <xs:restriction base=”xs:integer”>
                        <xs:minInclusive value=”10”/>
                        <xs:maxInclusive value=”20”/>
                </xs:restriction>
        </xs:simpleType>
</xs:attribute>
```

Einfacher, vorgegebener, benannter Typ

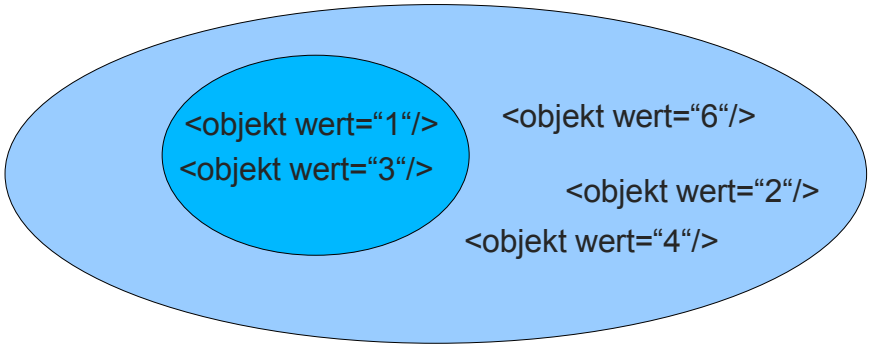
Einfacher, adhoc-definierter Typ (nur hier verwendet, unbenannt)

Typen und Ableitung

- Ableiten durch
  - Erweitern: Das Objekt bekommt neue Eigenschaften (heißt hier: neue Kindelemente oder Attribute)
  - Einschränken: Die Klasse enthält nicht mehr alle ursprünglichen Elemente
- Gleiches Konzept: Teilmengenbeziehung



Jedes Objekt im kleinen Kreis besitzt ein <e1>
(aber auch noch ein <e2>)



Jedes Objekt im kleinen Kreis hat einen ganzzahligen Wert
(aber nur die ungeraden)

Typen und Ableitung

- Beispiel

```
<xs:complexType name="personName">
   <xs:element name="title" minOccurs="0"/>
   <xs:element name="forename" minOccurs="0" maxOccurs="*"/>
   <xs:element name="surname"/>
</xs:complexType>
```

```
<xs:complexType name="extendedName">
  <xs:extension base="personName">
    <xs:sequence>
      <xs:element name="generation" minOccurs="0"/>
      <xs:element name="gender" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:extension>
</xs:complexType>
```

```
<xs:complexType name="simpleName">
  <xs:restriction base="personName">
    <xs:sequence>
      <xs:element name="title" maxOccurs="0"/>
      <xs:element name="forename" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:restriction>
</xs:complexType>
```

Typen und Ableitung

- Beispiel
  - Bislang definiert: <xsd:element name="Preis" type="decimal"/>
  - Neu definiert werden soll ein Typ für das folgende Element: <internationalerPreis waehrung="EUR">423.46</internationerPreis>
- Bilder der Ableitung mit <extension>
  - Hier: Erweiterung des Inhaltstyps „decimal“ zu „decimal mit Attribut 'waehrung' im Element“ → daher complexType

```
<xsd:element name="internationalerPreis">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:decimal">
        <xsd:attribute name="waehrung" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

Hinweis: Frühere Versionen von XML-Schema erzeugten Ableitungen durch das Attribut „derivedBy“; dieses Attribut ist in der aktuellen „Recommendation“ nicht mehr vorhanden

Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE schoko SYSTEM "http://meinserver/schoko.dtd">

<schoko
      xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
      xsi:noNamespaceSchemaLocation='schoko.xsd'>

        <schokolade nummer="1">
                <geschmack>Noisette</geschmack>
                <gewicht einheit=”g”>100</gewicht>
                <haltbar_bis>2005-07-01</haltbar_bis>
                <preis>0.79</preis>
        </schokolade>
        <schokolade nummer="2">
                <geschmack>Zartbitter</geschm^ack>
                <gewicht>100</gewicht>
                <haltbar_bis>2005-10-03</haltbar_bis>
                <preis>1.50</preis>
        </schokolade>
</schoko>
```

referenziert folgende DTD

referenziert folgendes XML-Schema

Für die Verwendung in Dokumenten mit Namensräumen gibt es entsprechend ein Attribut „schemaLocation“

Schema (1)

```
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="schoko">
    <xs:complexType>
       <xs:sequence>
          <xs:sequence>
(1) ->      <xs:element name="schokolade" maxOccurs="unbounded">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="geschmack" type="xs:string"/>
(2) ->            <xs:element name="gewicht">
                    <xs:complexType>
(3) ->              <xs:simpleContent>
(4) ->                <xs:extension base="xs:decimal">
                        <xs:attribute name="einheit" type="xs:string" use="optional" default="g"/>
                      </xs:extension>
                    </xs:simpleContent>
                  </xs:complexType>
                </xs:element>
```

Schema (2)

(1)

```
                <xs:element name="haltbar_bis" type="xs:date"/>
                <xs:element name="preis" type="xs:float"/>
              </xs:sequence>
              <xs:attribute name="nummer" type="xs:decimal"/>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

Erklärungen:

(1) Das Attribut von <schokolade> wird innerhalb des <complexType> definiert. Merke: Ein Element mit einem Attribut besitzt immer einen ComplexType!
(2) Das Element <gewicht> hat Attribute, ist also ein ComplexType.
(3) Eigentlich ist der Inhalt von <gewicht> ein einfacher Typ (SimpleType), nämlich xs:decimal. Da aber <gewicht> durch das Attribut kein SimpleType mehr sein kann, muss man den Typ ableiten!
(4) Man bildet also einen neuen Typ „Dezimal-und-Attribut-einheit“. Ein Element des Typs „dezimal-und-Attribut-einheit“ hat als Inhalt eine Dezimalzahl und muss selbst ein Attribut „einheit“ aufweisen.

DTD

(1)

```
<!ELEMENT schoko (schokolade)*>
<!ATTLIST schoko xmlns:xsi CDATA #IMPLIED>
<!ATTLIST schoko xsi:noNamespaceSchemaLocation CDATA #IMPLIED>

<!ELEMENT schokolade (geschmack, gewicht, haltbar_bis, preis)>
<!ATTLIST schokolade nummer CDATA #REQUIRED >
<!ELEMENT geschmack (#PCDATA)>
<!ELEMENT gewicht (#PCDATA)>
<!ATTLIST gewicht einheit CDATA “g”>
<!ELEMENT haltbar_bis (#PCDATA)>
<!ELEMENT preis (#PCDATA)>
```

Erklärung

(1) Wir müssen die zusätzlichen Attribute auch erklären, die für die Validierung mit XML-Schema gedacht sind. Wenn man keine XML-Schema-Validierung vorsieht, kann man in der XML-Datei die entsprechenden Angaben weglassen und entsprechend diese Attribute hier streichen.

XML-Verarbeitung

- Nichtvalidierender Parser
  - prüft nur die Wohlgeformtheit des XML-Dokuments
- Validierender Parser
  - prüft die Gültigkeit eines XML-Dokuments
- Ergebnis des Parsens
  - Baum, welcher die XML-Knoten beinhaltet: **Document Object Model (DOM)**
    oder
  - Kette von Ereignissen (Betreten eines Knotens, Daten...): **Simple API for XML (SAX)**

Document Object Model

- Objektorientierter Ansatz zur Analyse eines Dokuments
- Hierarchie von Dokumentknoten
  - Dokument, Element, Attribut, Text, Bemerkung
- DOM-API ist sprachunabhängig
  - Node: **firstChild/lastChild, previousSibling/nextSibling, childNodes,... insertBefore, replaceChild, ...**
  - Element (spezieller Node): **getElementsByTagName**,...
- Nachteil
  - Gesamtes Dokument muss eingelesen werden
  - speicherintensiv

SAX

- Meldet Ereignisse beim Durchlaufen des Baums
- Vorteil
  - XML-Strom-Verarbeitung, einfach und schnell
  - muss keinen Baum im Speicher aufbauen
  - ermöglichst das teilweise Parsen eines (großen) Dokuments
  - Uninteressantes kann ignoriert werden
- Nachteil
  - Ausdrucksmöglichkeiten begrenzt
  - Anwendungen müssen ggf. selbst Parsebaum aufbauen

Stylesheet-Transformation (XSLT)

- XML selbst definiert keine Präsentation
  - Umwandlung nach HTML für Darstellung im Browser
  - Umwandeln in TXT
  - Umwandeln in ein anderes XML-Format
- Extensible Stylesheet Language Transformation (XSLT)
  - Teil der XSL-Spezifikation
  - Transformationssprache mit Elementen von funktionalen Programmiersprachen
  - Template-basiert und deklarativ
  - benutzt XPath zur Navigation auf dem Dokumentbaum

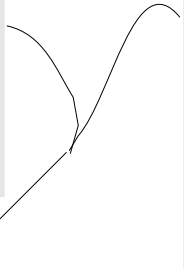
Beispiel

```
…
<bibliography>

<book>
  <title>Computer Networks</title>
  <author>A. Tanenbaum</author>
</book>

<book>
  <title>Distributed Systems</title>
  <author>G. Couloris</author>
</book>

</bibliography>
```



Titel	Autor
Computer Networks	A. Tanenbaum
Distributed Systems	G. Couloris

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://…" version="…">
<xsl:template match="/">

<html>
  <body>
    <table border="2">
      <tr>
        <th>Titel</th>
        <th>Autor</th>
      </tr>
      <xsl:for-each select="bibliography/book>
      <tr>
        <td> <xsl:value-of select="title"/></td>
        <td> <xsl:value-of select="author"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
</html>

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

→ **Übung**

XPath

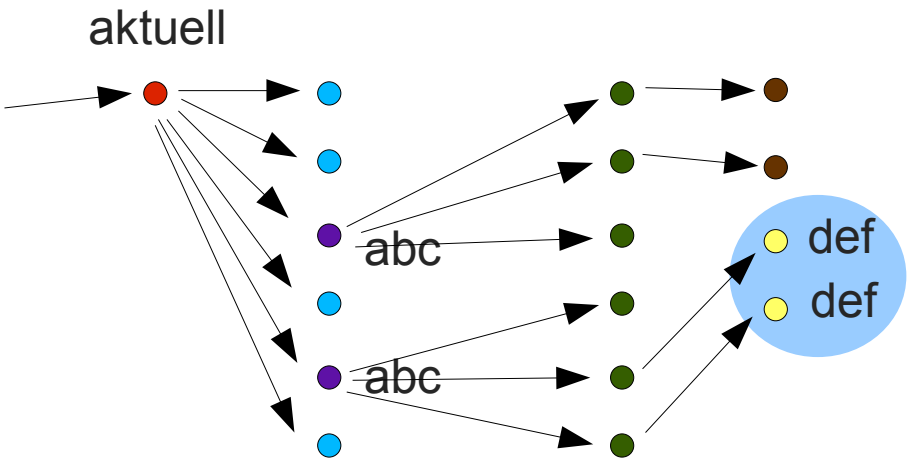
- Adressierung von Objekten im Dokument
- Sowohl einzelne Elemente/Attribute als auch Mengen

Beispieldokument für folgende XPath-Ausdrücke

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog type=“collection“>
  <cd country="USA">
    <title>Empire Burlesque</title><artist>Bob Dylan</artist><price>10.90</price>
  </cd>
  <cd country="UK">
    <title>Hide your heart</title><artist>Bonnie Tyler</artist><price>9.90</price>
  </cd>
</catalog>
```

XPath

- Spezielle Werte
  - abc referenziert alle Kindelemente namens abc
  - @abc: Attribut namens abc
  - * : alle Kindelemente
  - abc/*/def: alle Enkel namens „def“ jedes Kindknotens von Kindknoten namens „abc“ des aktuellen Knotens ... oder einfacher grafisch:



XPath

- Verkürzte Pfadangabe
  - abc//def: Alle Abkömmlinge namens „def“ jedes Knotens „abc“ (beliebige Tiefe, also abc/*/*/...*/def)
- Absolute Adressierung: / am Anfang
  - /catalog/cd: alle CDs
  - /cd = ø
  - //cd = /catalog/cd in diesem Beispiel (aber nicht allgemein)
- Indizierung (auch „Filterung“)
  - per Zahl (ab 1) oder „assoziativ“ mit XPath-Vergleichsausdruck
  - cd[3] = drittes Element „cd“
  - cd[@country=“USA“] = jenes Element „cd“ mit Wert „USA“ für Attribut „country“

XPath

- Vergleichsoperatoren =, !=, <, >, <=, >=
- Verknüpfung mit „and“ / „or“
- Beispiele
  - alter[@jahre] = alle Elemente „alter“, die ein Attribut „jahre“ haben
  - person[alter/@jahre<30] = jenes Element „person“, das einen Kindknoten namens „alter“ hat, dessen Attribut „jahre“ kleiner 30 ist
  - /catalog/@type=“collection“
  - //cd[@country=“UK“]
  - /catalog/cd[price>10]

Mittels XPath kann man Elemente, aber auch Attribute adressieren.
Daher referenziert //cd/@country=“USA“ die Menge aller Attribute namens USA innerhalb von „cd“-Knoten, aber nicht die Menge aller „cd“-Knoten mit diesem Attribut.

XML: Zusammenfassung

- XML: Extensible Markup Language
  - Sprache zur Definition anderer Sprachen (Metasprache)
  - Halbstrukturiertes Datenmodell
  - als Austauschsyntax verwendbar
  - textbasiert
- Dokumente werden als hierarchische Strukturen dargestellt
- Adressierbarkeit von Teilen des Dokuments
- Regeln für die Erzeugung von Dokumenten definierbar
- Große Unterstützung im Internet

Literatur

- XML-Schema
  - Einführung: http://www.w3.org/TR/xmlschema-0/
  - Strukturen: http://www.w3.org/TR/xmlschema-1/
  - Datentypen: http://www.w3.org/TR/xmlschema-2/
- Validator (DTD und XML-Schema)
  - http://www.validome.org/xml/validate/