

UNIT 0x10

XML

Lernziele

- **Herkunft bzw. Vorgänger von XML kennen**

Beides ist wichtig, da viele XML-Dokumente (auch die XML-Spezifikation selbst) Bezug darauf nehmen

- **Syntax von XML kennen**

XML-Dateien sind strukturierte Text-Dateien. Sie sollen die Strukturelemente und den Aufbau eines gültigen (wohlgeformten) XML-Dokumentes kennen.

- **Anwendungsgebiete für XML kennen**

XML wird in vielen Bereichen eingesetzt, hat dort aber einen anderen Namen

- **XML im Ansatz bewerten können**

Sie sollten Vor- und Nachteile von XML kennen



Historie

- **XML = eXtensible Markup Language**
- **Ausgangspunkt: Textsatz**
- **Schreibmaschinen-Zeitalter:**

Manuskripte wurde auf Schreibmaschine getippt
Anweisungen wie „Fettdruck“, „Kursiv“ an Setzer als handschriftliche Kommentare (bzw. als *Markup* - engl. für Textauszeichnung, abgeleitet von „*marking up*“)
- **Computer-Zeitalter:**

zunächst wie Schreibmaschine - nach Ausdruck handschriftliche *Markups*
Prozess trägt Möglichkeiten der EDV aber nicht Rechnung!
daher: spezielle *Markups* in Text integriert (formalisierte *Markup*)

Historie

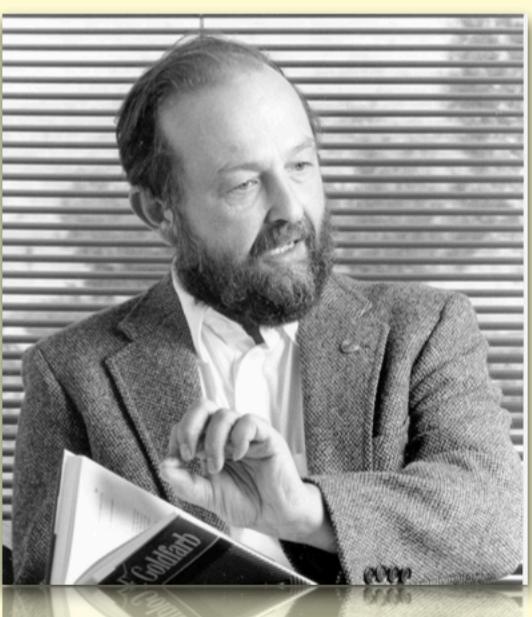
Jahrzehnt	Dokument-Formate	Programmierung	Netzwerke
60er 70er	Generalized Markup Language (GML) von IBM (Goldfarb, Mosher and Lorie)	Pascal, C, Simula 67	TCP/IP (Kahn und Cerf)
Lochkarten (80 Spalten), Assemblerprogrammierung, Bildschirme mit 80x12 Zeichen			

GML
:h1. Erstes Kapitel: Einführung
:p. GML unterstützt hierarchische Container, z.B.
:ol
:li. geordnete Listen,
:li. ungeordnete Listen,
:li. Definitionslisten
:eol.
 sowie einfache Strukturen.
:p. Die vereinfachte Auszeichnung erlaubt es, abschließende Formatierungsbefehle wie zum Beispiel für die Elemente "h1" oder "p" auszulassen.

Strikte Trennung von

- **was** (Überschrift, Absatz, Aufzählung etc.) und
- **wie** (Art der Darstellung, z.B. Font, Zeilenabstand etc.)
- GML wurde von der *Document Composition Facility* (eine Art Compiler) in IBM's Druckersprache SCRIPT/VS übersetzt
- **Profile bestimmen, wie Dokument schlussendlich dargestellt wird**
Profile gab es z.B. für Laser- und Nadeldrucker sowie diverse Bildschirme

Historie



Charles F. Goldfarb

Übergang zum Deskriptiven Markup

*That project required integrating a **text editing** application with an **information retrieval** system and a **page composition** program. The **documents** had to be kept in a repository from which they **could be selected by queries**. The selected documents could be revised with the text editor and returned to the data base, or rendered for presentation by the composition program.*

*Standard stuff for SGML systems today, perhaps, but far from the way most people thought about document processing in 1969. So far, in fact, that the applications we needed to integrate were not only not designed to work together, they couldn't even run on the same operating system. Fortunately, we had access to CP-67, an early hypervisor that allowed multiple concurrent operating systems to run on the same computer and share files. The problem was that, even when Ed Mosher, Ted Peterson, and I finally got the programs to talk to one another, we found they each required different **procedural markup** in the document files.*

Quelle: <http://www.sgmlsource.com/history/roots.htm>

Prozedurales vs. Descriptives Markup

- **Prozedurales Markup**

Domain Specific Language (DSL) für den Textsatz. Instruktionen an einen „Word-Processor“ sind im Text eingebettet; Prozeduren, Schleifen und Makros sind oft möglich. Typische Beispiele sind **TeX** und **Postscript**.

- **Deskriptives Markup**

Ein bestimmter Text-Abschnitt wird mit einem *Markup* markiert, es wird nicht festgelegt, wie er konkret verarbeitet werden soll. Dadurch erreicht man eine **Entkopplung der Gliederung eines Dokumentes von seiner konkreten Darstellung**. Ein Typisches Beispiel ist HTML, wo die konkrete Darstellung in ein CSS-Dokument aus gegliedert wird.

Historie

Jahrzehnt	Dokument-Formate	Programmierung	Netzwerke
80er	ISO-Standard Generalized Markup Language (SGML) TeX (Donald E. Knuth)	OOP, C++, GUIs	BITNET, EARN, CSNET, NSF-NET

PCs lösen Großrechner ab; Festplattenspeicher; Grafische Benutzeroberflächen (WYSIWIG)

```

<!-- Copyright (c) 2011 by Heini. This is free software. -->
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
<article>
  <sect1 id="introduction" lang="en">
    <title>Introduction & Basics</title>
    <para>
      Hello world!
    </para>
  </sect1>
</article>

```

SGML

SGML beschreibt Syntax auf zwei Ebenen

- **Allgemeine Syntax** - Prinzipieller Aufbau von Dokumenten
- **Spezielle Syntax** - Document-Type-Definition (DTD)

Legt fest, welche Tags mit welchen Attributen erlaubt sind und wie diese geschachtelt werden dürfen

Historie

Jahrzehnt	Dokument-Formate	Programmierung	Netzwerke
90er	MS-Word als de facto Standard, SGML nur Insidern bekannt	JAVA, komponenten-orientierte Programmierung	WWW wird Killer-applikation und ersetzt <i>ressource discovery systems</i> (Suchsysteme), Industrie entdeckt Internet:
HTML (Tim Berner-Lee): einfache Handhabung von Dokumenten im Netz			

```
<!-- Copyright (c) 2011 by Heini Hein. This is free software. -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Introduction</title>
  </head>
  <body>
    <h1>Introduction</h1>
    <p align="left">This introduction covers the following topics:
      <ul>
        <li>XML
        <li>HTML
      </ul>
    Enjoy!
    </p>
  </body>
</html>
```

HTML

HTML ist eine Anwendung von SGML!

Historie

Jahrzehnt	Dokument-Formate	Programmierung	Netzwerke
2000er	XML (1998) wird die technische Basis einer Integration von Dokumenten, Medien, Datenverarbeitung und Kommunikationsnetzen		

```

<?xml version="1.0"?>
<!-- Copyright (c) 2011 by Heini Hein. This is free software. -->
<!DOCTYPE MyFancyDoc SYSTEM "http://www.my.org/MyFancyDoc.dtd">
<hochschule name="FH Aachen">
  <fachbereiche>
    <fachbereich name="Medizintechnik und Technomathematik">
      <studiengaenge>
        <studiengang>
          <name>Scientific Programming</name>
          <grad>B.Sc.</grad>
          <dauer>6</dauer>
        </studiengang>
        <studiengang>
          <name>Technomathematik</name>
          <grad>M.Sc.</grad>
          <dauer>4</dauer>
        </studiengang>
      </studiengaenge>
    </fachbereich>
  </fachbereiche>
</hochschule>

```

XML

XML ist Vereinfachung von SGML

Zentrale Anwendungen für XML

- **Stetig wachsende Nutzung des Internet für e-Commerce**
 - Business-to-Business (B2B)
 - Business-to-Computer (B2C)
- **Dadurch: zunehmende Bedeutung des elektronischen Datenverkehrs**
 - Hier hat sich XML zum de facto-Standard entwickelt
- **Datenquelle: meist Relationale Datenbankmanagementsysteme**
- **Zwei Techniken für uns besonders relevant**
 - **Definition und Darstellung von Sichten auf eine Datenbank mit XML**
Technologien: Document Type Definition (DTD) und XML-Schema
 - **Definition eines kompletten Datenbank-Schemas in XML**
Technologien: XPath und XQuery

Standard

- **Herausgeber ist das *Word Wide Web Consortium (W3C)***
 - W3C spricht von „Recommendation“, nicht von Standard
- **Zwei Versionen sind relevant**
 - Version 1.0 vom 28.11.2008 - siehe <http://www.w3.org/TR/2008/REC-xml-20081126/>
Auf diese Version stützt sich diese Vorlesung
 - Version 1.1 vom 16.08.2006 - siehe <http://www.w3.org/TR/2006/REC-xml11-20060816/>
Änderungen sind eher marginal und werden hier nicht behandelt
- **Spezifikation beschreibt**
 - Klasse von Datenobjekten (XML-Dokumente) und
 - teilweise das Verhalten eines XML-Prozessors, einem Software-Modul zum Lesen von XML-Dokumenten und zum Zugriff auf dessen Inhalt und Struktur

Syntax - Überblick

- Processing Instructions
- Kommentare
- Tags
- Text
- Entity-Referenzen
- weiteres (später)

```
<?xml version="1.0"?>
<!-- Copyright (c) 2011 by Heini. --&gt;
&lt;hs&gt;
  &lt;fachbereiche&gt;
    &lt;fb&gt;
      &lt;name&gt; Medizintechnik &amp; Technomathematik &lt;/name&gt;
      &lt;studiengaenge&gt;
        &lt;studiengang&gt;
          &lt;name&gt; Scientific Programming &lt;/name&gt;
          &lt;grad&gt; B.Sc. &lt;/grad&gt;
          &lt;dauer&gt; 6 &lt;/dauer&gt;
        &lt;/studiengang&gt;
        &lt;studiengang&gt;
          &lt;name&gt; Technomathematik &lt;/name&gt;
          &lt;grad&gt; M.Sc. &lt;/grad&gt;
          &lt;dauer&gt; 4 &lt;/dauer&gt;
        &lt;/studiengang&gt;
      &lt;/studiengaenge&gt;
    &lt;/fb&gt;
  &lt;/fachbereiche&gt;
&lt;/hs&gt;</pre>
```

Syntax – Tags

- Ein XML-Dokument enthält **Tags**
- Tags beginnen mit "<" und enden auf ">"; man unterscheidet
 - öffnende Tags; z.B. `<fachbereich>`
 - schließende Tags; z.B. `</fachbereich>`
 - Kurztags; z.B. `<fachbereich/>`
- Tag-Namen dürfen aus
 - Buchstaben,
 - Ziffern,
 - Minuszeichen, Punkt, Doppelpunkt und Unterstrichen bestehen;
- Tags müssen mit einem
 - Buchstaben,
 - Unterstrich oder
 - Doppelpunkt beginnen
- Umlaute sind erlaubt -> Unicode!

**Exakte Festlegungen für gültige Namen in XML
finde Sie hier: [http://www.w3.org/TR/2008/
REC-xml-20081126/#sec-common-syn](http://www.w3.org/TR/2008/REC-xml-20081126/#sec-common-syn) (s.n.F)**

Namensregeln

- Hier zum Nachschlagen die XML-Namensregel

```
NameStartChar ::= ":" | [A-Z] | "_" | [a-z] | [#xC0-#xD6] | [#xD8-#xF6] |
[ #xF8-#x2FF] | [#x370-#x37D] | [#x37F-#x1FFF] |
[#x200C-#x200D] | [#x2070-#x218F] |
[#x2C00-#x2FEF] | [#x3001-#xD7FF] | [#xF900-#xFDCF] |
[#xFDF0-#xFFFFD] | [#x10000-#xEFFFF]
NameChar      ::= NameStartChar | "-" | "." | [0-9] | #xB7 |
[#x0300-#x036F] | [#x203F-#x2040]
Name          ::= NameStartChar (NameChar)*
```

Tipp: Eine Übersicht über alle Unicode-Zeichen finden Sie hier:
<http://www.utf8-zeichentabelle.de/>

Syntax – Tags

- **Tags sind case-sensitive;**
 - d.h. `` und `` sind unterschiedliche Tags
- **Tags dürfen nicht mit xml beginnen** (auch nicht mit XML, xMI etc.)

```
<xml-test>Ein Text</xml-test>
```



- **Zu jedem öffnenden Tag existiert ein passendes schließendes Tag**
 - in HTML erlaubt!

```
<ul>  
  <li>Eins  
  <li>Zwei  
</ul>
```



```
<p>Erster Abschnitt  
<p>Zweiter Abschnitt  
</p> <!-- Welcher wird beendet? -->
```



- **Öffnende und schließende Tags sind korrekt balanciert**
 - in HTML erlaubt!

```
<b>  
  <it>fett und kursiv  
  </it>  
</b>
```



Syntax – Elemente

- Als **Element** bezeichnet man den Bereich vom öffnenden zum schließenden Tag (einschl. der Tags selbst)
- Der Name des öffnenden Tags ist der **Typ** des Elements
- Ein XML-Dokument besteht aus verschachtelten Elementen:

```
<studiengaenge>
    <studiengang>
        <name>Scientific Programming</name>
    </studiengang>
</studiengaenge>
```

The XML code is shown with some parts highlighted in yellow boxes. The outermost tag, `<studiengaenge>`, has the word "studiengaenge" written twice to its right. The inner tag, `<studiengang>`, has the word "studiengang" written once to its right. Inside the `<studiengang>` tag, the `<name>` tag contains the text "Scientific Programming", which is highlighted in a yellow box, and has the word "name" written once to its right.

- Der Bereich zwischen den Tags wird **Inhalt** genannt
- Kurtztags sind folglich Elemente ohne Inhalt
- Der Inhalt eines Elements besteht aus Elementen oder Daten
 - Ein Element beschreibt daher einen Baum

XML-Dokument als Baum

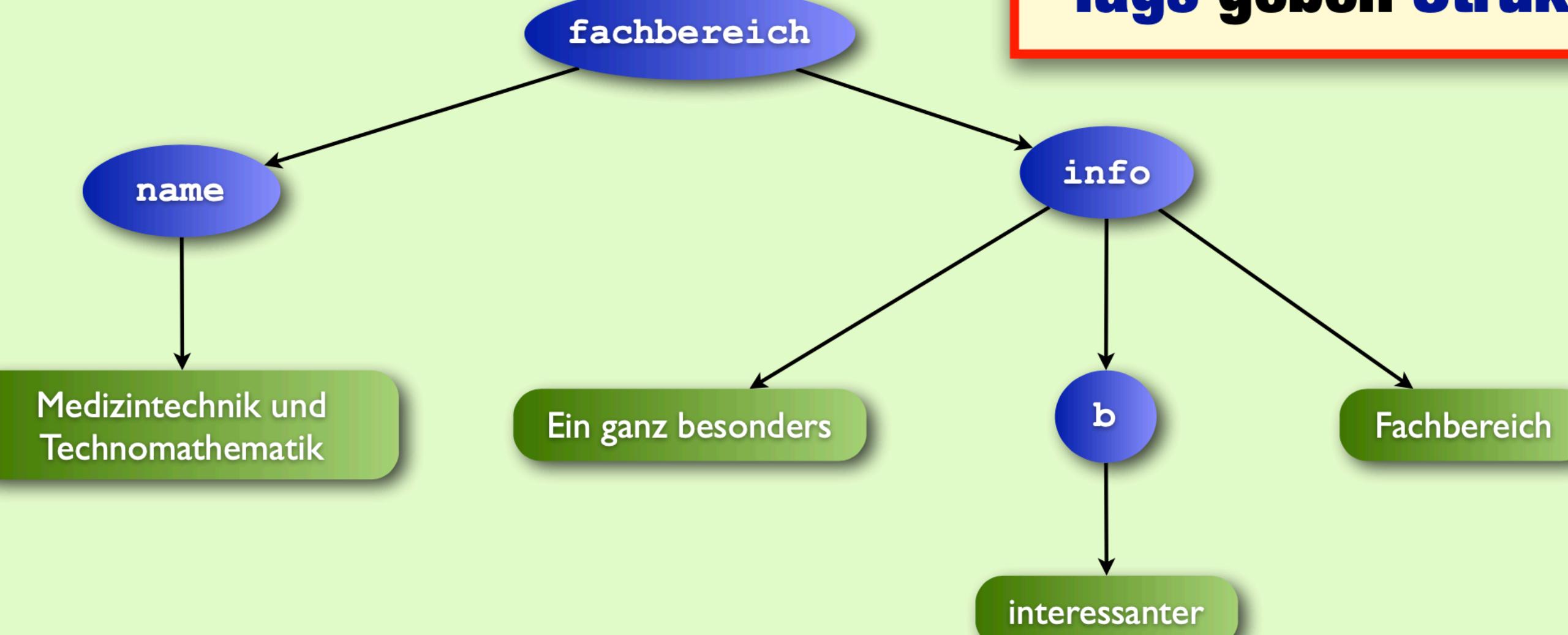
B XML-Element als Baum (XML-Baum)

XML-Fragment:

```
<fachbereich>
  <name>Medizintechnik und Technomathematik</name>          Inhalt
  <info>Ein ganz besonders <b>interessanter</b> Fachbereich</info>
</fachbereich>
```

Zugehöriger Baum:

Tags geben Struktur!



XML-Dokument als Baum

- **Wohlgeformte XML-Dokumente besitzen genau ein Element - das Wurzel-Element oder auch Dokumenten-Element**
 - Ein XML-Dokumente repräsentiert daher einen Baum (XML-Baum)
 - Auch Graphen sind möglich (über „Pointer“ - dazu später mehr)
- **An den Blättern eines XML-Baumes stehen**
 - **Kurztags** (leere Elemente),
 - **Text** (*PCDATA - parsed character data - analysierter Text*) oder
 - **CDATA-Knoten** (nicht analysierter Text)

Syntax – Text

- **Textknoten enthalten beliebigen Text**
- **Aber: Symbole „<“ und „&“ sind nicht erlaubt stattdessen müssen Entity-Referenzen verwendet werden**
 - **Entities** sind Makro-Definitionen; Entity-Referenzen verweisen darauf
 - Entity-Referenzen werden mit "&" eingeleitet und enden auf ";" z.B. wird "&" zum kaufmännischen Und "&" expandiert
 - Benutzerdefinierte Entities sind möglich (dazu später mehr)

Vordefinierte Entities	
Entity	Zeichen
amp	&
gt	>
lt	<
quot	"
apos	'

D Entity-Referenzen

`<name>Medizintechnik & Technomathematik</name>`

Ist illegal, da XML-Prozessor „&“ als Start einer Entity-Referenz interpretiert. Korrekt ist:

`<name>Medizintechnik &#amp; Technomathematik</name>`

- **Zeichenreferenzen verweisen auf spezifisches Unicode-Zeichen**
 - `0` für das Zeichen „0“ bzw.
 - `2`, falls der Zeichencode dezimal angegeben werden soll

Syntax - CDATA

- **Der XML-Prozessor zerlegt eine XML-Datei in lexikalische Bestandteile - Inhalt von Elementen wird daher immer analysiert**
 - Dieser kann Mixtur aus Text, weiteren Elementen und Entity-Referenzen sein
 - Man spricht von **PCDATA** (*parsed character data*)
- **CDATA-Knoten werden vom XML-Prozessor niemals zerlegt**
- **Sie starten mit "<! [CDATA[" und enden auf "]]>"**
 - Text zwischen diesen Markierungen wird 1:1 übernommen
 - "<" und "&" dürfen in CDATA-Knoten direkt verwendet werden
 - Text darf lediglich die Symbolfolge "]]>" nicht enthalten

Syntax - CDATA

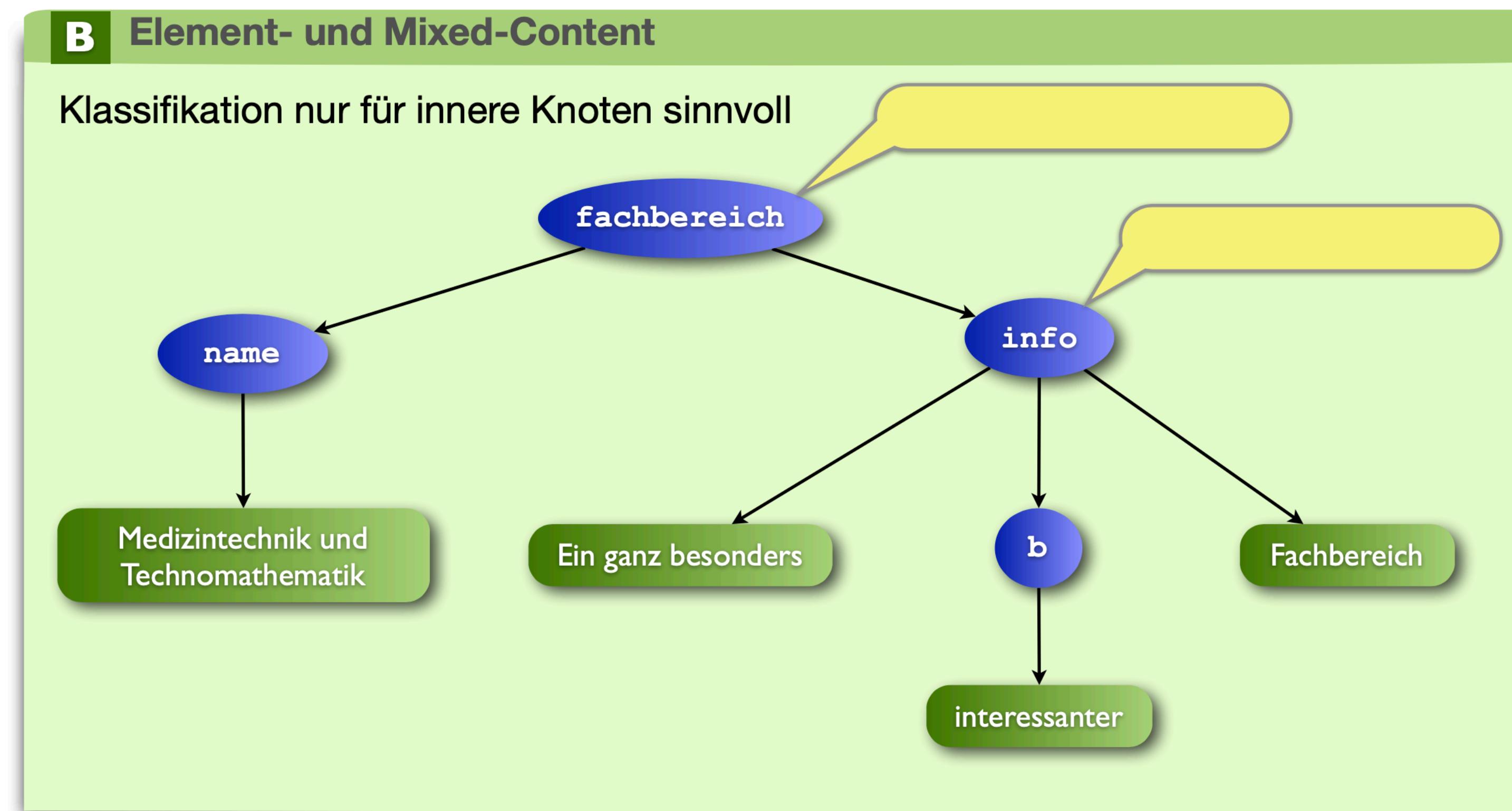
- Beispiel für einen CDATA-Knoten

B CDATA

```
<algorithm>
  <name>Bubble Sort</name>
  <implementation language="JAVA">
    <![CDATA[
      public void bubbleSort(int feld[]) {
        for (int i=feld.length-1 ; i>=0 ; --i)
          for (int j=1; j<=i ; ++j)
            if (feld[j]<feld[j-1]) {
              int t = feld[j] ; feld[j]=feld[j-1] ; feld[j-1]=t;
            }
      }
    ]]>
  </implementation>
</algorithm>
```

Inhalt von Elementen: Klassifikation

- Je nach Art der Kindknoten unterscheidet man Elemente mit
 - **Element Content**: Alle Kind-Knoten sind Element-Knoten
 - **Mixed Content**: Kind-Knoten sind Element-, Text- oder CDATA-Knoten



Syntax – Attribute

- Elemente können mit **Attributen** versehen werden
 - Definitionen innerhalb des öffnenden Tags bzw. des Kurztags
 - Format:
 - **Attributname="Wert"** oder
 - **Attributname='Wert'**
 - nicht erlaubt ist Mischen von " und ' ; z.B. **Attributname='Wert'" X**
 - Entity-Referenzen innerhalb der Werte sind zulässig (ggf. notwendig!)
 - Attributname darf nur einmal verwendet werden
 - Attribute sind ungeordnet

B Attribute

```
<fachbereich name="Medizintechnik & Technomathematik" nr="9">
  <dekan name="Volker Sander" raum="O1G31"/>
  <prodekane>
    <prodekan name='Andreas Terstegge' />
    <prodekan name="Karl Ziemons" raum="O1F19" />
  </prodekane>
</fachbereich>
```

Attribut oder Element?

- **Daten besser als Attribut**

```
<dekan name="Volker Sander" raum="O1G31" />
```

oder als Element

```
<dekan>
  <name>Volker Sander</name>
  <raum>=O1G31</raum>
</dekan>
```

speichern?

- **Keine offizielle Festlegung durch XML-Standard, aber Hinweise:**

Attribut	Element
ungeordnet (Element-Knoten besitzt Attributmenge)	geordnet (Element-Knoten hat mehrere Element-Knoten als Kinder)
1:1 - Beziehung (Ein Attribut nur einmal pro Element)	1:n - Beziehung (Element kann wiederholt als Kindknoten auftreten)
Wert unstrukturiert (einfacher Text)	Wert strukturiert (Baum)

Reservierte Attributnamen (Beispiele)

- **xml:lang**
 - Sprache des Inhalts
 - Beispiel: `<p xml:lang="de">Aufgabe 1</p>`
- **xml:space**
 - Leerräume im Inhalt B
 - Beispiel: `<p xml:space="[preserve/default]">Aufgabe 1</p>`
- **xml:id**
 - Elementbezeichner (dokumentweit eindeutig)
 - Beispiel: `<p xml:id="Sektion_1">Sektion 1</p>`
- **xml:base**
 - Basis-URL (für relative Links)
 - Beispiel: `<ul xml:base="http://www.example.com/docs/2012/">`
`XML-Attribute`
``

Leere Elemente?

B Leere Elemente

```
<name>
  <first>Donald</first>
  <last>Knuth</last>
</name>
```

vs.

```
<name>
  <first>Donald</first>
  <middle/>
  <last>Knuth</last>
</name>
```

- **leere Elemente können Attribute haben; z.B.**

```
<name>
  <first>Donald</first>
  <middle status="unknown"/>
  <last>Knuth</last>
</name>
```

- **evtl. von DTD oder XML-Schema vorgeschrieben (dazu später)**

Syntax – PIs

- **Procssing-Instructions sind Anweisungen an einen XML-Prozessor**
 - von daher u.U. spezifisch für bestimmte Implementierungen eines Prozessors
- **XML-Dokument sollte mit „XML-Instruction“ beginnen**
 - sie gibt Auskunft über die verwendete **Version** von XML und die **Zeichenkodierung**
 - ist die Zeichenkodierung weder UTF-8 noch UTF-16, so ist die XML-Instruction Pflicht!

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```
 - für XML 1.1 obligatorisch!
- **xmlstylesheet ist eine weitere Processing Instruction**
 - mit ihr können stylesheets zur Darstellung von xml-Dateien spezifiziert werden
 - wird hier nicht behandelt!

Syntax – Kommentare

- **Kommentare beginnen mit „<!--“ und enden auf „-->“**

- dürfen nicht vor der XML-Instruction stehen
- sind nur ausserhalb des Markups erlaubt

```
<name <!-- Ein Name --> >Heinrich</name>
```

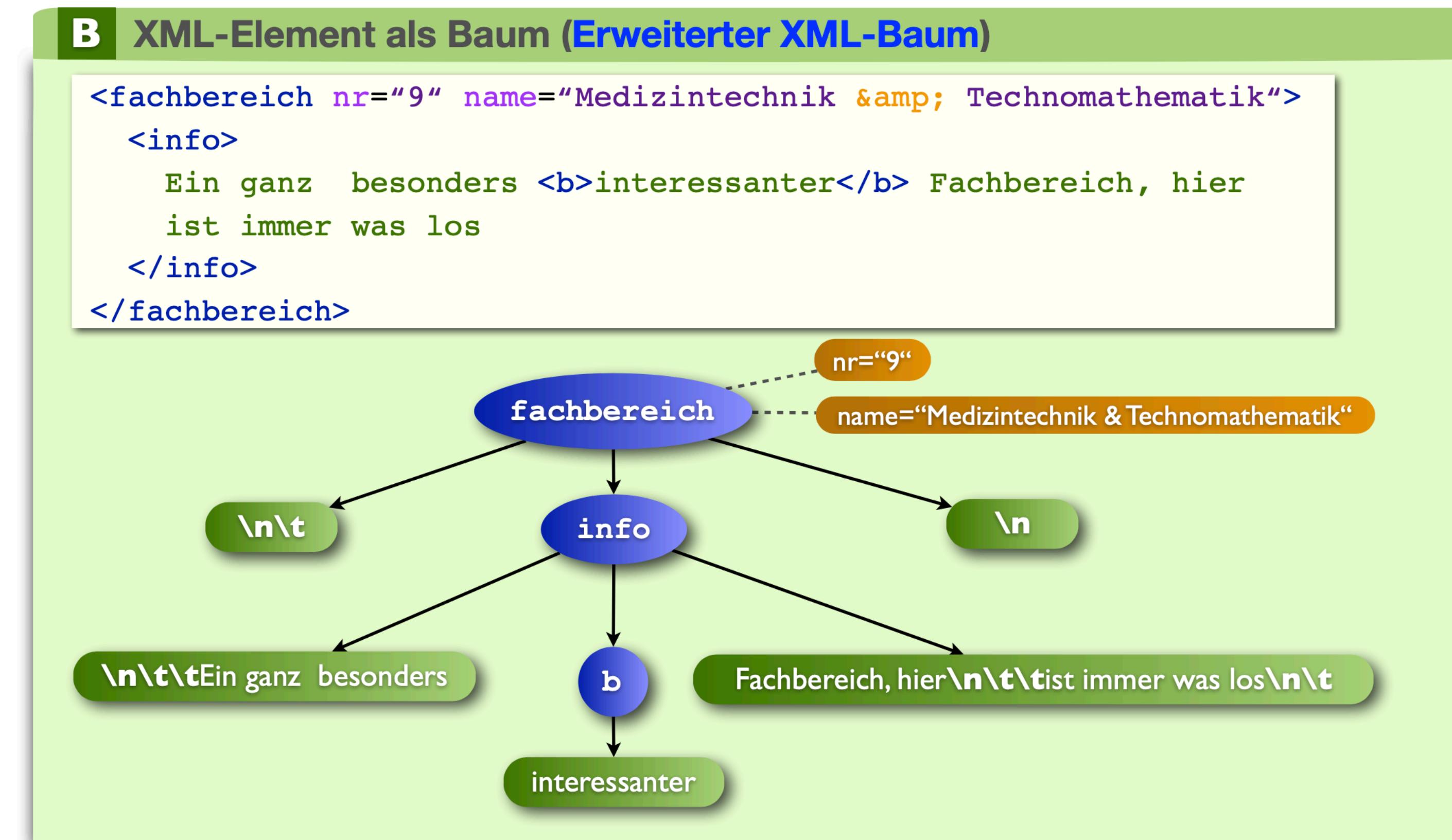


- dürfen die Zeichenfolge „--“ nicht enthalten
- Entity-Referenzen werden nicht expandiert - „<“ und „&“ als Zeichen sind erlaubt

```
<!--  
Ein Kommentar darf sich über mehrere  
Zeilen erstrecken. & wird nicht zu ✓  
& ersetzt.<ja!>  
-->
```

XML und Whitespaces

- **Whitespaces** bleiben in XML erhalten
 - Whitespaces sind: Leerzeichen (#x20), Tabulator (#x9), Zeilenende (#xA) und Zeilenvorschub (#xD)



XML und Zeilenende

- **Kodierung des Zeilenendes variiert von Betriebssystem zu Betriebssystem:**
 - UNIX: #xA (Zeilenvorschub)
 - OSX (Mac): #xD (Wagenrücklauf)
 - DOS und Windows: #xD#xA (Wagenrücklauf und Zeilenvorschub)
- **XML-Prozessor reicht an Anwendung nur Zeilenvorschub #xA weiter!**
 - Vorsicht! Wurde in XML 1.1 geändert

Wohlgeformtes XML

- Ein Dokument beinhaltet wohlgeformtes XML (*well-formed XML*) wenn es die XML-Syntaxregeln respektiert; insbesondere muss gelten:
 - Es gibt genau ein Wurzel-Element
 - Element- und Wurzelnamen entsprechen den Festlegungen in <http://www.w3.org/TR/2008/REC-xml-20081126/#sec-common-syn>
 - Zu jedem Start-Tag gibt es ein korrespondierendes End-Tag
(Beachte XML ist case-sensitive)
 - Elemente überlappen sich nicht (korrekte Schachtelung)
 - Ein Element enthält keine Attribute mit identischen Namen
 - Jedes Attribut hat einen Wert

Flexibilität

- Dokumente, die sich an obige Syntaxregeln halten, nennt man **wohlgeformtes XML**
- Wohlgeformtes XML bildet nur einen groben Rahmen zur Darstellung von Daten (keine vordefinierten Elemente!)
- Um Daten wieder auffindbar bearbeiten zu können reichen diese Festlegungen nicht aus, denn
 - Man muss wissen, wo was steht - dafür muss Dokument-Struktur bekannt sein!
 - Man muss wissen, wie die Daten an den Blättern kodiert sind - dafür braucht man so etwas wie Daten-Typen!
- Reines XML oft Basis für andere Sprachen
 - spezielle Sprachen schränken XML bewusst ein (z.B. durch kontextfreie Grammatik)
 - Später: DTDs und XML-Schema -> XML als **Meta-Sprache**

Anwendungen von und mit XML

- **Primär: Datenaustausch**
- **Textsatz (klassische, von SGML „geerbt“)**
 - DocBook
 - Hier auch MathML (mathematische-) und CML (chemische Formeln)
- **Word Wide Web**
 - XHTML als konsistenter Ersatz von HTML
 - SOAP - *Simple Object Access Protocol* - Nachrichtenaustausch zwischen Anwendungen
 - RSS - *Really Simple Syndication*
- **Austausch von Unternehmensinformationen** (z.B. Jahresabschlüsse)
 - XBRL - *Extensible Business Reporting Language*
- **Katalogisierung von Büchern, CDs etc.**
 - Dublin Core
- **Kodierung von Grafiken und Filmen**
 - SVG - *Scalable Vector Graphics* und SMIL - *Synchronized Multimedia Integration Language*
- **Dokumente konvertieren**
 - XSLT - *Extensible Stylesheet Language Transformations*
 - XSL-FO - *XSL Formatting Objects*
- **Konfiguration von Software (z.B. Hibernate, Android)**
- **Abfragen von XML-Dokumenten**
 - XPATH, X-Query

Bewertung – positiv

- **XML-Dateien sind Textdateien**
 - mit einfachen Mitteln (Editor) lesbar
 - eventuell selbsterklärend (falls Namen von Tags und Attributen klaren Hinweis auf Semantik geben)
- **XML ist flexibel**
 - Eigene „Datensprache“ oder Adaption einer bestehenden
 - Einfache Erweiterung existierender Sprachen (z.B. Hinzufügen von Elementen oder Attributen)
- **XML eignet sich gut für den Datenaustausch**
 - XML ist ein offener Standard und lässt sich leicht transformieren
- **XML erfreut sich einer umfangreicher Werkzeugunterstützung**
 - Validierende XML Prozessoren (Parser), XSLT-Prozessoren, XML-Editoren etc.
 - Diverse Programmierbibliotheken bis hin zur vollständigen Integration in Programmiersprachen (z.B. via JAVA-Annotationen)

Bewertung - negativ

- **Konflikte zwischen Spezifikation**
 - insbesondere z.B. XML-Schema vs. Relax NG vs. DTD
 - Welche Technologie einsetzen?
- **Spezifikationen gelten z.T. als zu kompliziert**
 - z.B. XML-Schema-Spezifikation des W3C oder ODF - das Open Document Format
- **Integration in Browser noch nicht perfekt**
- **XML birgt Overhead**
 - **Speicherplatz**: klar, daher werden XML-Dateien häufig gepackt
 - **Laufzeit**: XML-Dateien müssen in interne Programmstrukturen (z.B. Objektgraphen) überführt werden

GESCHAFFT...

PROF. DR. RER. NAT. ALEXANDER VÖß

FH AACHEN
UNIVERSITY OF APPLIED SCIENCES