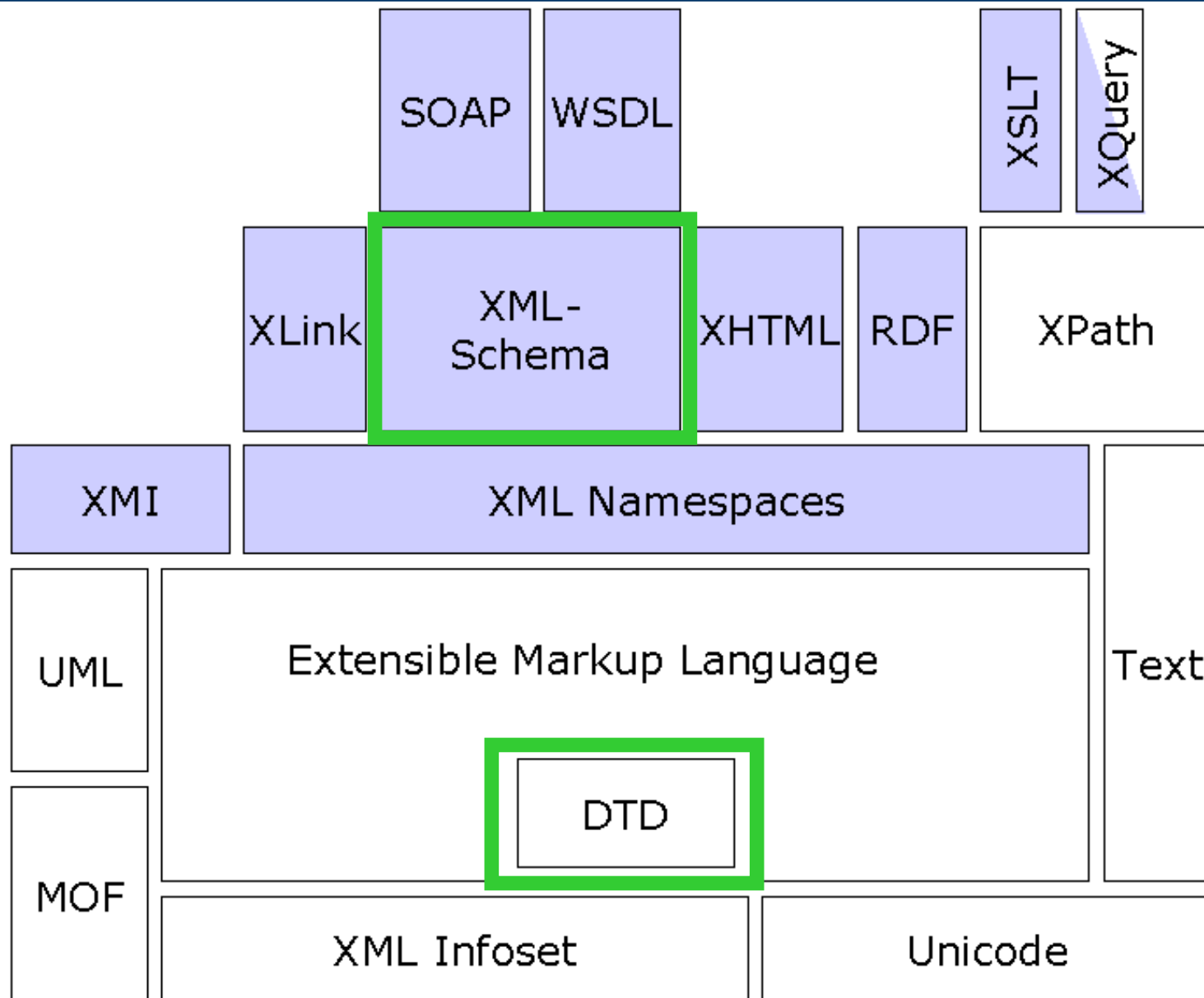




Document Type Definitions (DTDs)

Robert Tolksdorf
Freie Universität Berlin
Institut für Informatik
Netzbasierende Informationssysteme
tolk@ag-nbi.de



Quelle: <http://www.jeckle.de/images/xml/languageFamily.gif>

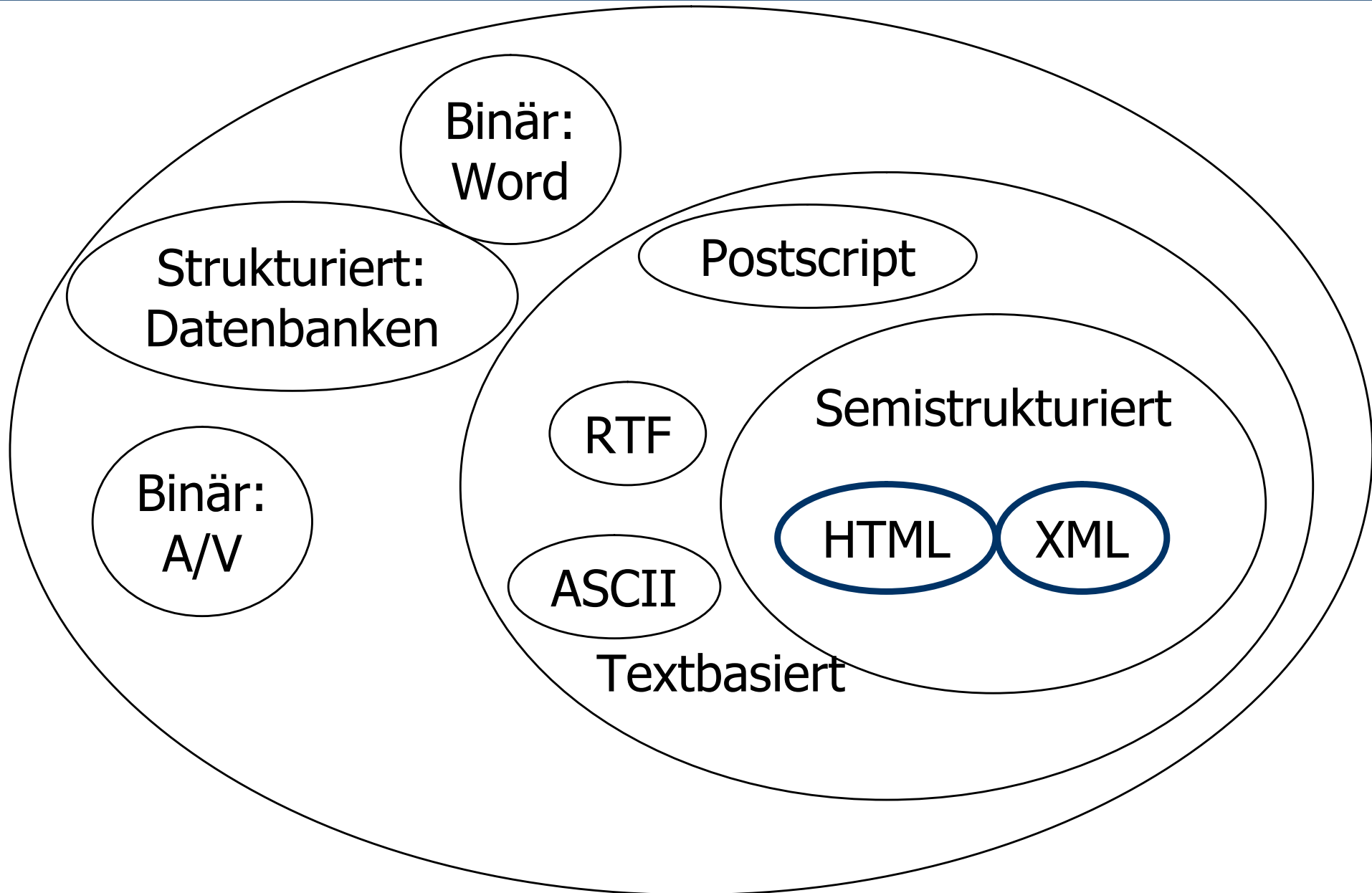
oder so?

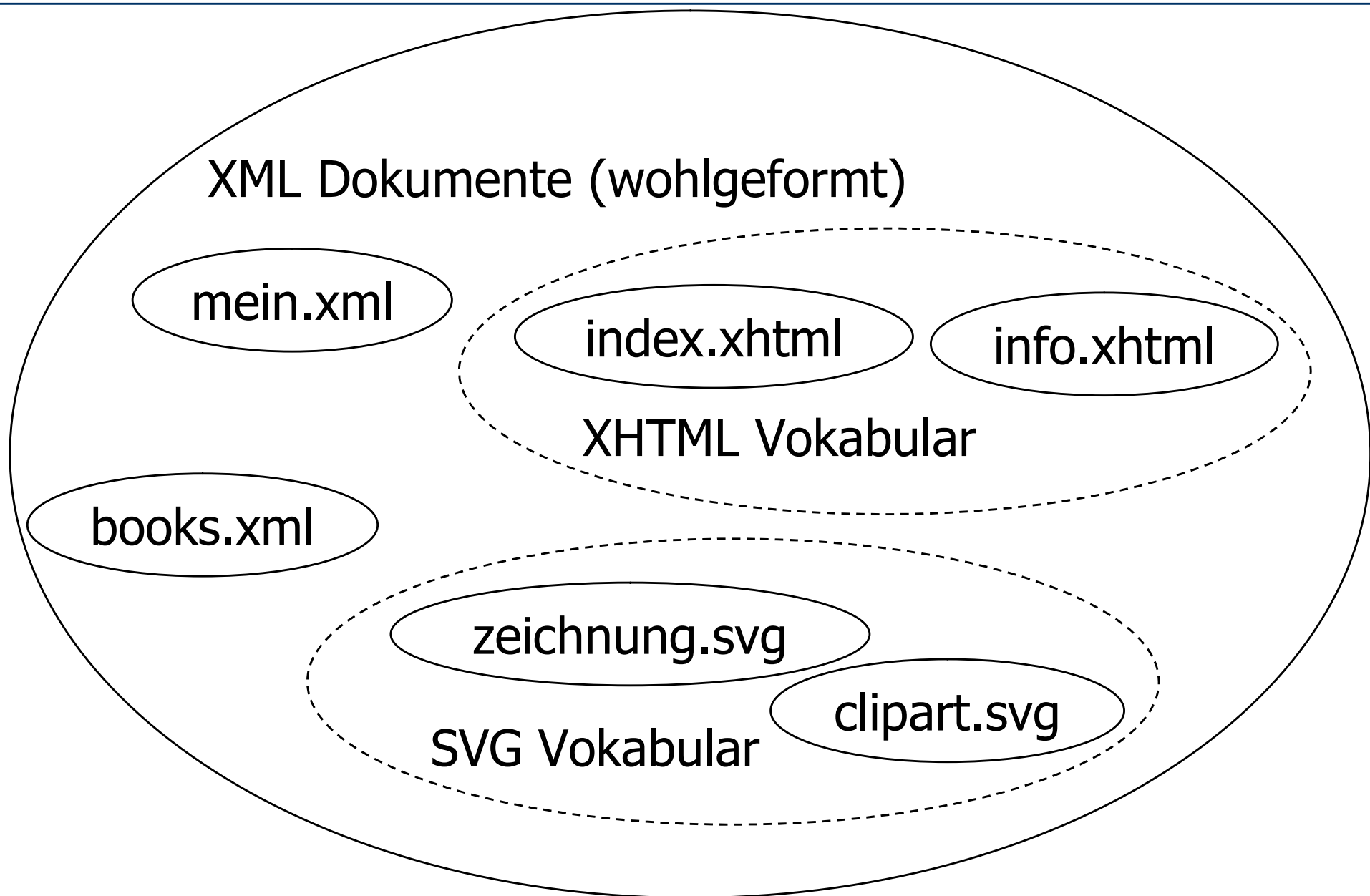
```
<book>
  <title>My Life and Times</title>
  <authors>
    <author>
      <first>Paul</first>
      <last>McCartney</last>
    </author>
  </authors>
  <date>
    <year>1998</year>
    <month>July</month>
  </date>
  <isbn>94303-12021-43892</isbn>
  <publisher>McMillin
Publishing</publisher>
</book>
```

```
<Book>
  <Title>My Life and Times</Title>
  <Author>Paul McCartney</Author>
  <Date>July, 1998</Date>
  <ISBN>94303-12021-43892</ISBN>
  <Publisher>McMillinPublishing</Publisher>
</Book>
```

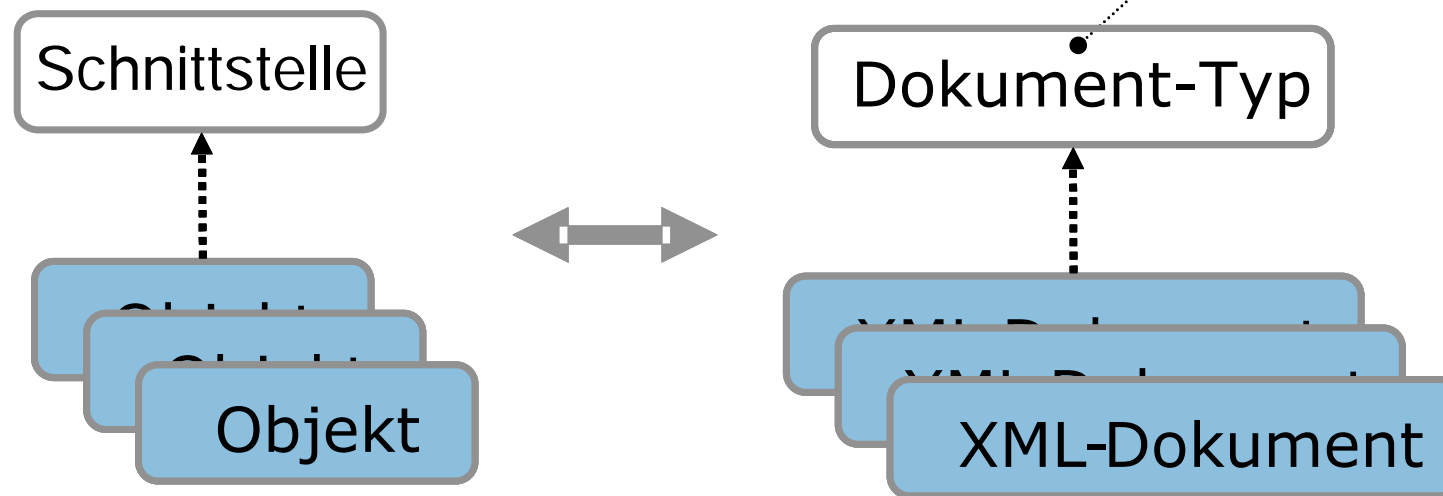
so?

- ⇒ einheitliches Format nötig
- ⇒ Format sollte durch XML-Prozessor validierbar sein





Typ von erlaubten XML-Dokumenten = "XML-Sprache"



- Dokument-Typ definiert mit einer DTD, einem XML-Schema oder ähnlichen Formalismen

- **prinzipieller Aufbau von Dokumenten:**

Welche Elemente/Attribute?

- **Datentypen der Inhalte:**

Welche Inhalte?

```
<Book>  
  <Title> PCDATA </Title>  
  <Author> PCDATA  
  </Author>  
  <Date> PCDATA </Date>  
  <ISBN> PCDATA </ISBN>  
  <Publisher> PCDATA  
  </Publisher>  
</Book>
```

- konkrete Inhalte werden nicht beschrieben

⇒ Klasse von erlaubten XML-Dokumenten



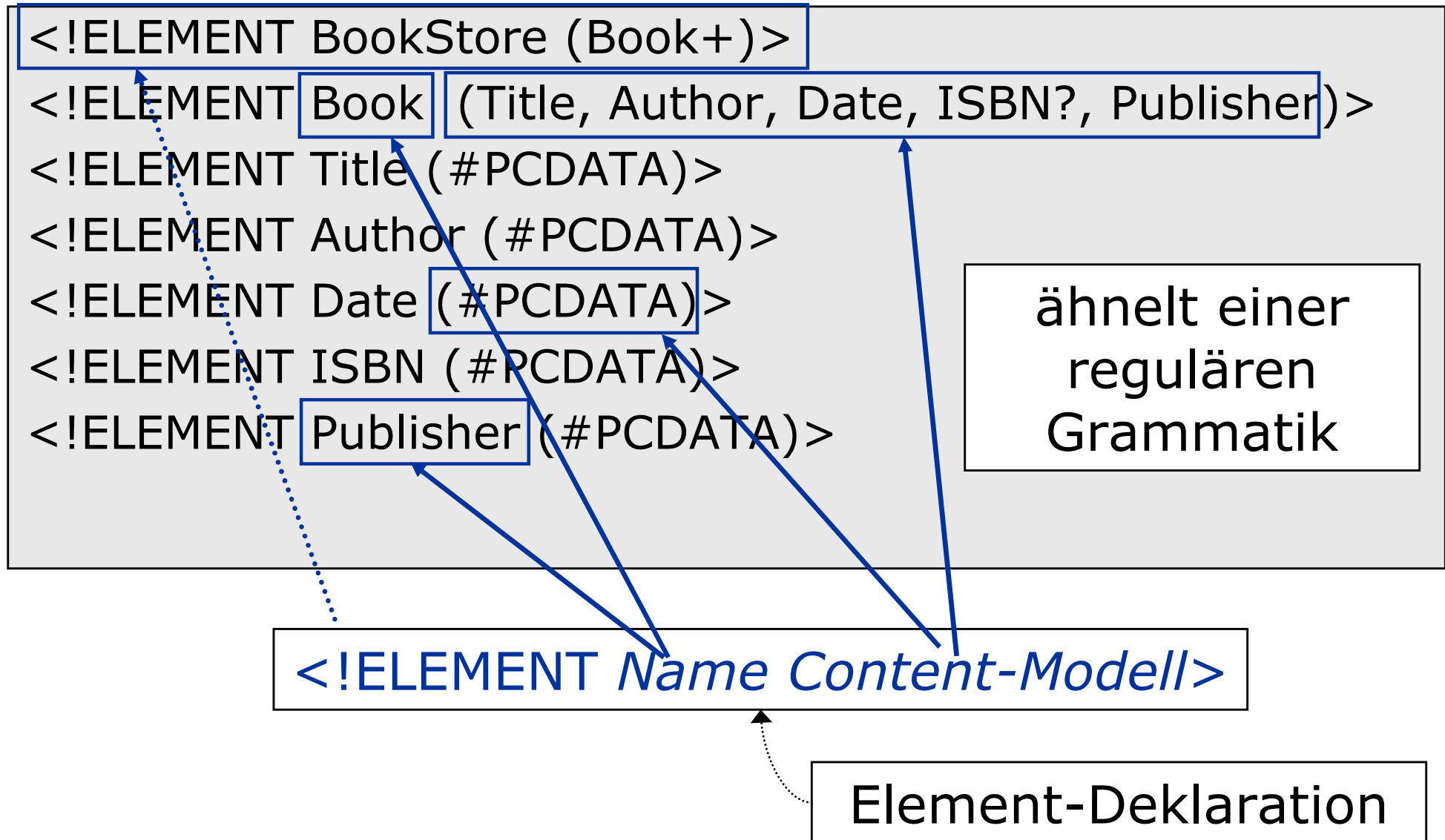
Document Type Definitions (DTDs)

Wie sieht eine DTD hierfür aus?

```
<BookStore>
  <Book>
    <Title>My Life and Times</Title>
    <Author>Paul McCartney</Author>
    <Date>July, 1998</Date>
    <ISBN>94303-12021-43892</ISBN>
    <Publisher>McMillin Publishing</Publisher>
  </Book>
</BookStore>
```

- BookStore soll mindestens ein Buch enthalten.
- ISBN optional
- alle anderen Kind-Elemente obligatorisch

Die DTD für das Beispiel-Dokument



- Element:
Ausdruck über Elemente mit Symbolen , + * | ?
- **#PCDATA**:
unstrukturierter Inhalt ohne reservierte Symbole (<,&)
<!ELEMENT Title (#PCDATA)>
- **EMPTY**:
leerer Inhalt, Element kann Attribute haben
<!ELEMENT hr EMPTY>:<hr height="3"/>
- **ANY**:
beliebiger Inhalt (strukturiert, unstrukturiert, gemischt oder leer)
<!ELEMENT Absatz ANY>
- Keine gewohnten Datentypen wie INTEGER oder FLOAT

```
<!ELEMENT BookStore (Book+)>
```

```
<BookStore>  
  <Book>...</Book>  
  <Book>...</Book>  
</BookStore>
```

- **+**: n Wiederholungen mit $n > 0$.
- *****: n Wiederholungen mit $n \geq 0$.

- BookStore hat mindestens ein Kind Book
- Außer Book darf BookStore keine anderen Kind-Elemente haben.

```
<!ELEMENT BookStore (Book | (Book, BookStore))>
```

- Bookstore besteht aus genau einer der Alternativen:
 - genau ein Kind-Element Book
 - zwei Kind-Elemente: Book und BookStore
- | : Auswahl, genau eine der beiden Alternativen
- , : Sequenz von Elementen.
- Beachte: Rekursive Deklaration nicht äquivalent zur vorherigen, iterativen Definition!

Rekursive vs. iterative Deklaration

```
<BookStore>  
  <Book>...</Book>  
  <Book>...</Book>  
  <Book>...</Book>  
</BookStore>
```

BookStore mit 3
Büchern

```
<!ELEMENT BookStore (Book+)>
```

```
<BookStore>  
  <Book>...</Book>  
  <BookStore>  
    <Book>...</Book>  
    <BookStore>  
      <Book>...</Book>  
    </BookStore>  
  </BookStore>  
</BookStore>
```

BookStore mit 3
Büchern

```
<!ELEMENT BookStore (Book | (Book, BookStore))>
```

```
<!ELEMENT BookStore (Book+)>
```

```
<!ELEMENT Book (Title, Author, Date, ISBN?, Publisher)>
```

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

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

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

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

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

```
<!ELEMENT Book (Title, Author, Date, ISBN?, Publisher)>
```

- Title, Author, Date, ISBN und Publisher (in dieser Reihenfolge) Kind-Elemente von Book
- außer diesen keine anderen Kind-Elemente
- ? : optional

```
<Book>  
  <Title>...</Title>  
  <Author>...</Author>  
  <Date>...</Date>  
  <ISBN>...</ISBN>  
  <Publisher>...</Publisher>  
</Book>
```



```
<!ELEMENT BookStore (Book+)>
```

```
<!ELEMENT Book (Title, Author, Date, ISBN?, Publisher)>
```

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

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

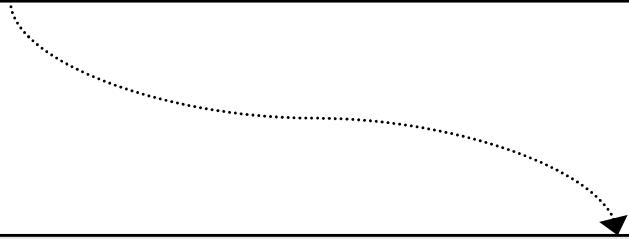
```
<!ELEMENT Date (#PCDATA)>
```

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

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

Deklaration von Title etc.

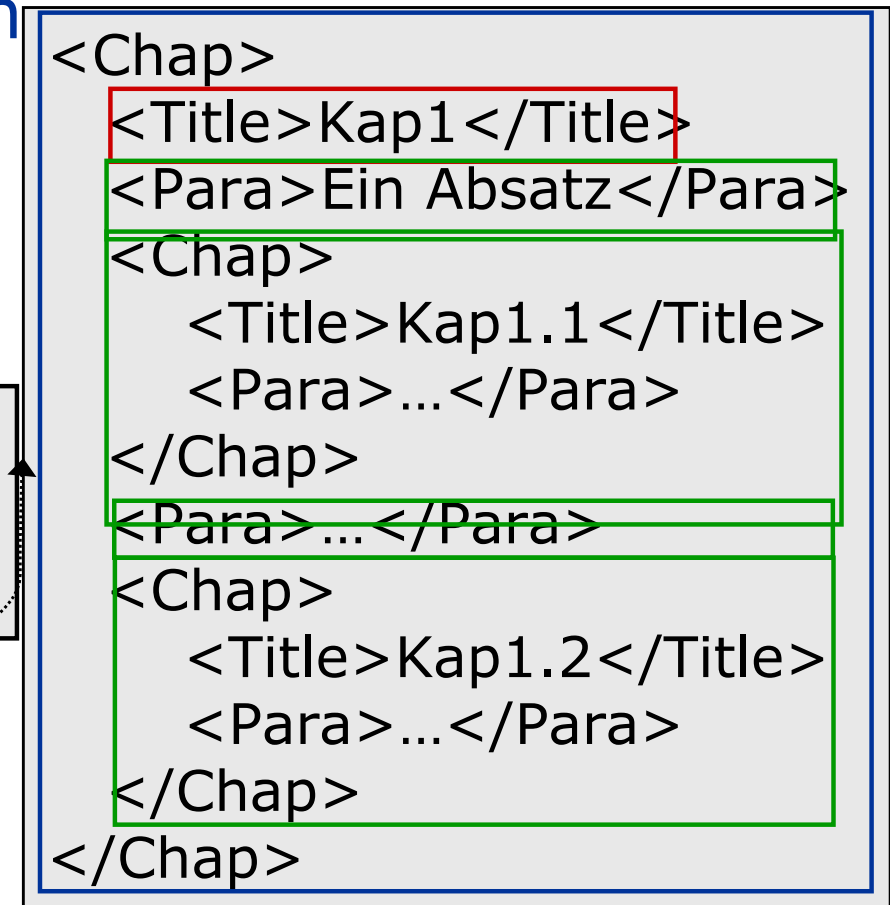
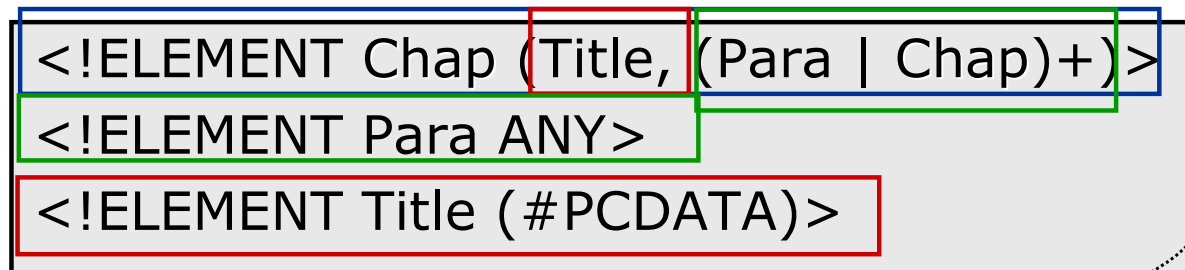
```
<!ELEMENT Title (#PCDATA)>  
<!ELEMENT Author (#PCDATA)>  
<!ELEMENT Date (#PCDATA)>  
<!ELEMENT ISBN (#PCDATA)>  
<!ELEMENT Publisher (#PCDATA)>
```



```
<Title>My Life and Times</Title>  
<Author>Paul McCartney</Author>  
<Date>July, 1998</Date>  
<ISBN>94303-12021-43892</ISBN>  
<Publisher>McMillin Publishing</Publisher>
```

Verschachtelungen

- beliebige Verschachtelung von Sequenz, Auswahl |, ?, *, + und Rekursion erlaubt
- Beispiel:



- Beispiel: $((b, c) \mid (b, d))$ ist **nicht-deterministisch**
- Grund: Wenn erstes Element = b, dann kann XML-Prozessor keine der beiden Alternativen ausschließen.
- XML erlaubt nur **deterministische** Content Modelle
- jedes nicht-deterministische Content Modell kann in ein äquivalentes deterministisches umgeformt werden
- Beispiel: $((b, c) \mid (b, d)) = (b, (c \mid d))$



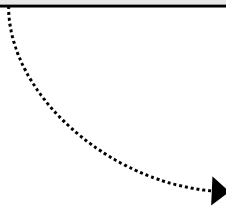
DTDs: Attribut-Deklaration

```
<!ATTLIST BookStore  
version CDATA #IMPLIED>
```

```
<!ATTLIST Name  
AttrName1 AttrTyp1 Attrbeschr1  
AttrName2 AttrTyp2 Attrbeschr2  
>
```

} Attribut-
Deklarationen

```
<!ATTLIST BookStore  
        version CDATA #IMPLIED>
```



```
<BookStore version="1.0">  
    ...  
</BookStore>
```

- BookStore hat Attribut version
- Außer version hat BookStore keine weiteren Attribute
- **CDATA**: Attribut-Wert = String ohne <, &, ', "
- Beachte: nicht verwechseln mit <![CDATA[...]]>
- daher Entity References für <, & und ' bzw. " verwenden

```
<!ATTLIST Author  
          gender (male | female) "female">
```


- hier statt CDATA **Aufzählungstyp**:
- Attribut gender hat entweder Wert male oder female.
- "female" ist Standard-Wert von gender.

Zusätzlich zu CDATA (Strings) und Aufzählungstypen:

- **NMTOKEN**: String, der Namenskonventionen von XML entspricht
- **ID**: eindeutiger Bezeichner, der Namenskonventionen von XML entspricht
- **IDREF**: Referenz auf einen eindeutigen Bezeichner

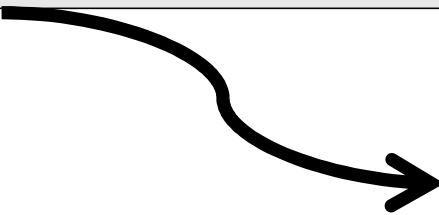
- Leerzeichen nicht zulässig
- eignet sich somit dazu Werte mit Leerzeichen auszuschließen

```
<!ELEMENT Datei EMPTY>  
<!ATTLIST Datei Name NMTOKEN #REQUIRED>
```



```
<Datei Name="readme.txt"/>
```

```
<!ELEMENT Woerterbuch (#PCDATA)>  
<!ATTLIST Woerterbuch ISBN NMTOKEN #REQUIRED>
```



```
<Woerterbuch ISBN="3-57710-446-5">  
Deutsches Wörterbuch  
</Woerterbuch>
```

Beispiel von: <http://www.maik-stuehrenberg.de/arbeit/projekte/milca/a-5-4/A-5-4-3-3-3-2-7.xhtml>

```
<!ATTLIST Author  
    key ID #IMPLIED  
    keyref IDREF #IMPLIED>
```

- Wert des Attributes key muss eindeutig sein:
Zwei Attribute vom Typ ID dürfen nie gleichen Wert haben
- Wert von keyref muss gültige Referenz sein:
keyref muss Wert eines Attributes vom Typ ID sein

```
<BookStore>
  <Book>
    <Title>Text</Title>
    <Author key="k1">Text</Author>
    <Date>Text</Date>
    <Publisher pkey="p1">Text</Publisher>
  </Book>
  <Book>
    <Title>Text</Title>
    <Author keyref="k1"/>
    <Date>Text</Date>
    <Publisher pkey="p1">Text</Publisher>
  </Book>
</BookStore>
```

Wert **k1** muss eindeutig sein:
kein anderes Attribut vom Typ
ID darf diesen Wert haben.

Referenz **k1** muss existieren:
ein Attribut vom Typ ID muss
den Wert **k1** haben.

Deklaration von Attributen

```
<!ATTLIST BookStore  
version CDATA #IMPLIED>
```

```
<!ATTLIST Name  
AttrName1 AttrTyp1 Attrbeschr1  
AttrName2 AttrTyp2 Attrbeschr2  
>
```

} Attribut-
Deklarationen

```
<!ATTLIST BookStore version CDATA #FIXED "1.0">
```

- **#FIXED**: Attribut hat immer den gleichen Wert
- **#IMPLIED**: Attribut optional
- **#REQUIRED**: Attribut obligatorisch



DTDs: Entitäten

XML Entitäten

- Vom letzten Mal:
 - Entity References in XML:
 - & ⇒ &
 - < ⇒ <
 - > ⇒ >
 - ' ⇒ '
 - " ⇒ "
- Entities sind Abkürzungen für Zeichenfolgen
- In DTDs selbst definierbar


```
<!ENTITY author "Robert Tolksdorf" >
```

```
<!ENTITY Name Definition >
```

- Verwendung: `<Fusszeile>Autor: &author</Fusszeile>`
- "Import" aus externen Definitionen:
`<!ENTITY fub SYSTEM "http://www.fu-berlin.de/defs.dtd" >`
`<!ENTITY c PUBLIC "-//W3C//TEXT copyright//EN"`
`"http://www.w3.org/xmlspec/copyright.xml" >`
- „General Entity“ zur Verwendung in Dokumenten

Deklaration von Entitäten

```
<!ENTITY % t "(#PCDATA)">
```

```
<!ENTITY % Name Definition >
```

- „Internal Entity“ zur Verwendung in DTDs
- Verwendung: `<!ELEMENT Fusszeile %t;>`
- „Import“ aus externen Definitionen:

```
<!ENTITY % fu-attrs SYSTEM "http://www.fu-berlin.de/defs.dtd">
```

```
<!ENTITY % Shape PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- `%fu-attrs;` übernimmt alle Definitionen aus der Datei



DTDs: Deklaration von Dokument-Typ

- vollständige DTD intern im XML-Dokument

```
<!DOCTYPE Wurzel-Element [...] >
```

- ein Verweis auf eine externe DTD im XML-Dokument

```
<!DOCTYPE Wurzel-Element SYSTEM "DTD" >
```

oder

```
<!DOCTYPE Wurzel-Element PUBLIC "DTD" "URL" >
```

- Dokument-Typ direkt nach XML-Deklaration einfügen

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BookStore [
  <!ELEMENT BookStore (Book+)>
  <!ELEMENT Book (Title, Author, Date, ISBN?, Publisher)>
  <!ELEMENT Title (#PCDATA)>
  <!ELEMENT Author (#PCDATA)>
  <!ELEMENT Date (#PCDATA)>
  <!ELEMENT ISBN (#PCDATA)>
  <!ELEMENT Publisher (#PCDATA)>
]>
<BookStore>
  ...
</BookStore>
```

`<!DOCTYPE Wurzel-Element SYSTEM "DTD">`

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BookStore SYSTEM "Bookstore.dtd">
<BookStore>
  ...
</BookStore>
```

```
<?xml version="1.0" encoding="UTF-16" ?>
<!DOCTYPE Book SYSTEM "Book.dtd">
<Book>
  ...
</Book>
```

Dokument-Typ
Deklaration



DTDs: Wohlgeformt & zulässig

- wohlgeformt (well formed):

XML-Dokument entspricht Syntaxregeln von XML

- zulässig (valid) bzgl. einer DTD:

1. Wurzel-Element des XML-Dokumentes in DTD deklariert
2. Wurzel-Element hat die in der DTD festgelegte Struktur



Ein Blick in die damalige DTD von XHTML

```
<!--
  Extensible HTML version 1.0 Strict DTD
  [...]
  Copyright (c) 1998-2002 W3C (MIT, INRIA, Keio),
  All Rights Reserved.

  This DTD module is identified by the PUBLIC and SYSTEM identifiers:

  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  SYSTEM "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"

  $Revision: 1.1 $
  $Date: 2002/08/01 13:56:03 $
-->

<!--===== Character mnemonic entities =====>

<!ENTITY % HTMLlat1 PUBLIC
  "-//W3C//ENTITIES Latin 1 for XHTML//EN"
  "xhtml-lat1.ent">
%HTMLlat1;
```

```
<!-- Portions (C) International Organization for Standardization 1986
      Permission to copy in any form is granted for use with
      conforming SGML systems and applications as defined in
      ISO 8879, provided this notice is included in all copies.
```

```
-->
```

```
[...]
```

```
<!ENTITY nbsp    " "> <!-- no-break space = non-breaking space,
                          U+00A0 ISOnum -->
```

```
<!ENTITY iexcl  "¡"> <!-- inverted exclamation mark, U+00A1 ISOnum -->
```

```
<!ENTITY cent   "¢"> <!-- cent sign, U+00A2 ISOnum -->
```

```
<!ENTITY pound  "£"> <!-- pound sign, U+00A3 ISOnum -->
```

```
<!ENTITY curren "¤"> <!-- currency sign, U+00A4 ISOnum -->
```

```
<!ENTITY yen    "¥"> <!-- yen sign = yuan sign, U+00A5 ISOnum -->
```

```
<!ENTITY brvbar "¦"> <!-- broken bar = broken vertical bar,
                          U+00A6 ISOnum -->
```

```
<!ENTITY sect   "§"> <!-- section sign, U+00A7 ISOnum -->
```

```
<!ENTITY uml    "¨"> <!-- diaeresis = spacing diaeresis,
                          U+00A8 ISODia -->
```

```
<!--===== Generic Attributes =====-->
```

```
<!-- core attributes common to most elements
```

```
  id      document-wide unique id  
  class   space separated list of classes  
  style   associated style info  
  title   advisory title/amplification
```

```
-->
```

```
<!ENTITY % coreattrs
```

```
"id      ID          #IMPLIED  
class    CDATA       #IMPLIED  
style    %StyleSheet; #IMPLIED  
title    %Text;      #IMPLIED"  
>
```

```
<!ENTITY % i18n
```

```
"lang     %LanguageCode; #IMPLIED  
xml:lang  %LanguageCode; #IMPLIED  
dir       (ltr|rtl)      #IMPLIED"  
>
```

```
<!ENTITY % attrs "%coreattrs; %i18n; %events;">
```

```
<!ENTITY % lists "ul | ol | dl">
```

```
<!ENTITY % block
```

```
  "p | %heading; | div | %lists; | %blocktext; | fieldset | table">
```

```
<!-- Unordered list -->
```

```
<!ELEMENT ul (li)+>
```

```
<!ATTLIST ul
```

```
  %attrs;
```

```
>
```

```
<!-- list item -->
```

```
<!ELEMENT li %Flow;>
```

```
<!ATTLIST li
```

```
  %attrs;
```

```
>
```

```
<!-- %Flow; mixes block and inline and is used for list items etc. -->
```

```
<!ENTITY % Flow "(#PCDATA | %block; | form | %inline; | %misc;)*">
```

Nachteile von DTDs

- keine XML-Syntax, eigener Parser nötig
- sehr wenige Datentypen
- keine eigenen Datentypen definierbar
- keine Namensräume:
DTDs können nur dann kombiniert werden, wenn es keine Namenskonflikte gibt!
- keine Vererbungshierarchien für Typen

Und noch ein Nachteil

- Sequenzen einfach zu definieren:
<!ELEMENT Book (Title, Author)>
- Aber: Soll Reihenfolge der Kind-Elemente egal sein, müssen alle Permutationen explizit aufgezählt werden:
<!ELEMENT Book ((Title, Length) | (Length, Title))>
- nicht praktikabel: bei n Kind-Elementen n! Permutationen

DTD deklariert das erlaubte Vokabular

DTD definiert für jedes Element ein Content-Modell

- Content-Modell legt fest:
 - Elemente oder Daten
 - Reihenfolge & Anzahl von Elementen/Daten innerhalb eines Elements
 - Pflicht oder optionale Elemente

DTD deklariert für jedes Element eine Menge von erlaubten Attributen