XML-Praxis

# XML – Extensible Markup Language

Jörn Clausen
joern@TechFak.Uni-Bielefeld.DE

# XML? Das sind doch bloß spitze Klammern!

XML? Das sind doch bloß spitze Klammern!

Ja, und Programme sind nur Folgen von Bytes.

# Übersicht

- Was ist XML?
- Verarbeitung von XML
- Aufbau von XML-Dokumenten
- XML-Grammatiken

# Was ist XML?

- Daten sind strukturiert (Texte, Bilder, Meßergebnisse, . . . )
- maschinelle Verarbeitung erfordert Kenntniss der Strukturen
- gesucht: Formalismus, um beliebige Strukturen zu beschreiben
- XML kann textuelle Daten strukturieren
- standardisierte Methoden zur Verarbeitung von XML

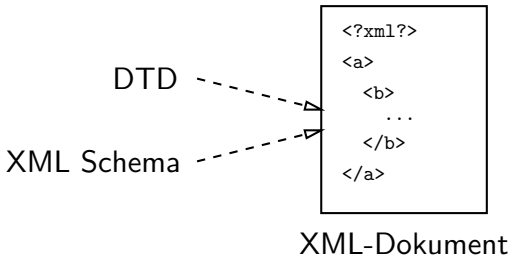
# Ursprung von XML: SGML

- Standard Generalized Markup Language
- 1986 als ISO-Standard 8879 verabschiedet
- *keine* Markup-Sprache, sondern Grammatik-Sprache
- Einsatz vor allem im Verlagswesen
- 1989: Hypertext Markup Language, World Wide Web
- Problem: komplexe Regeln, Parser schwer zu implementieren
- ab 1996: Entwicklung von XML
- SGML−−, nicht HTML++

# Verarbeitung von XML

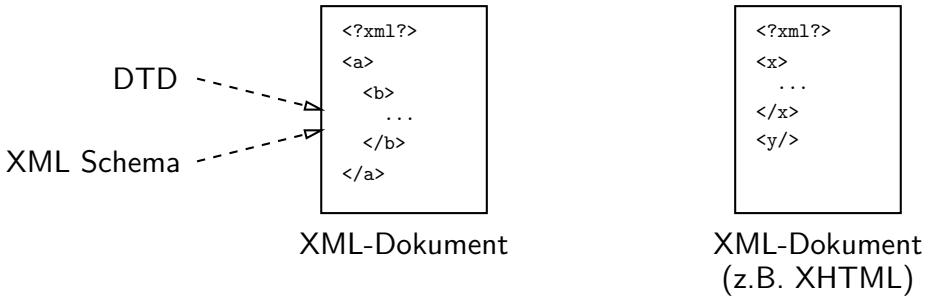

XML-Dokument

# Verarbeitung von XML



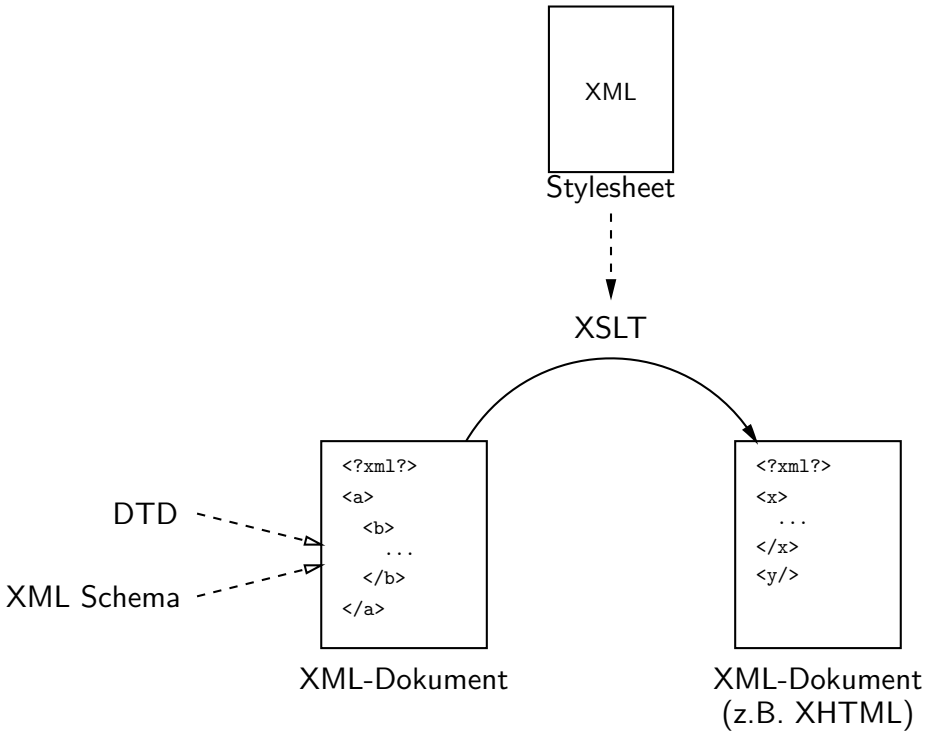

DTD: Document Type Definition

# Verarbeitung von XML



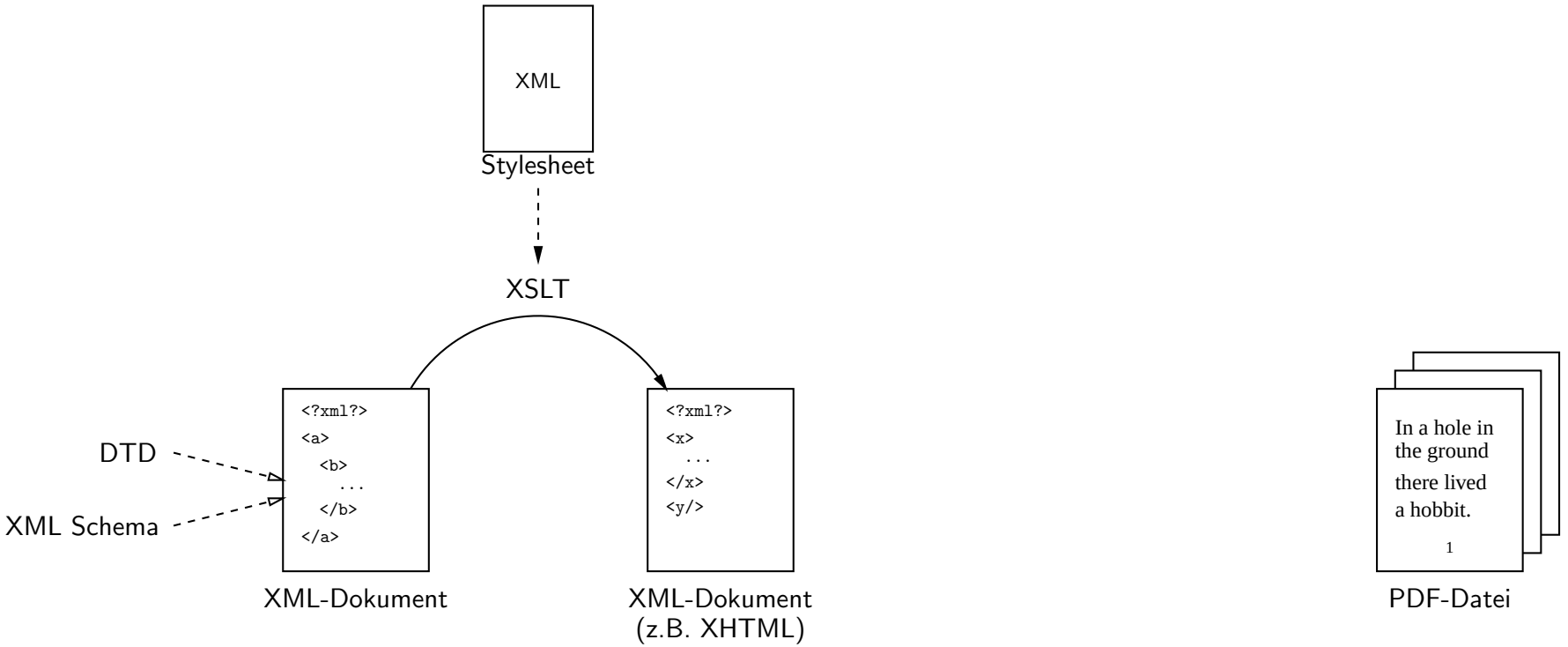

DTD: Document Type Definition

# Verarbeitung von XML



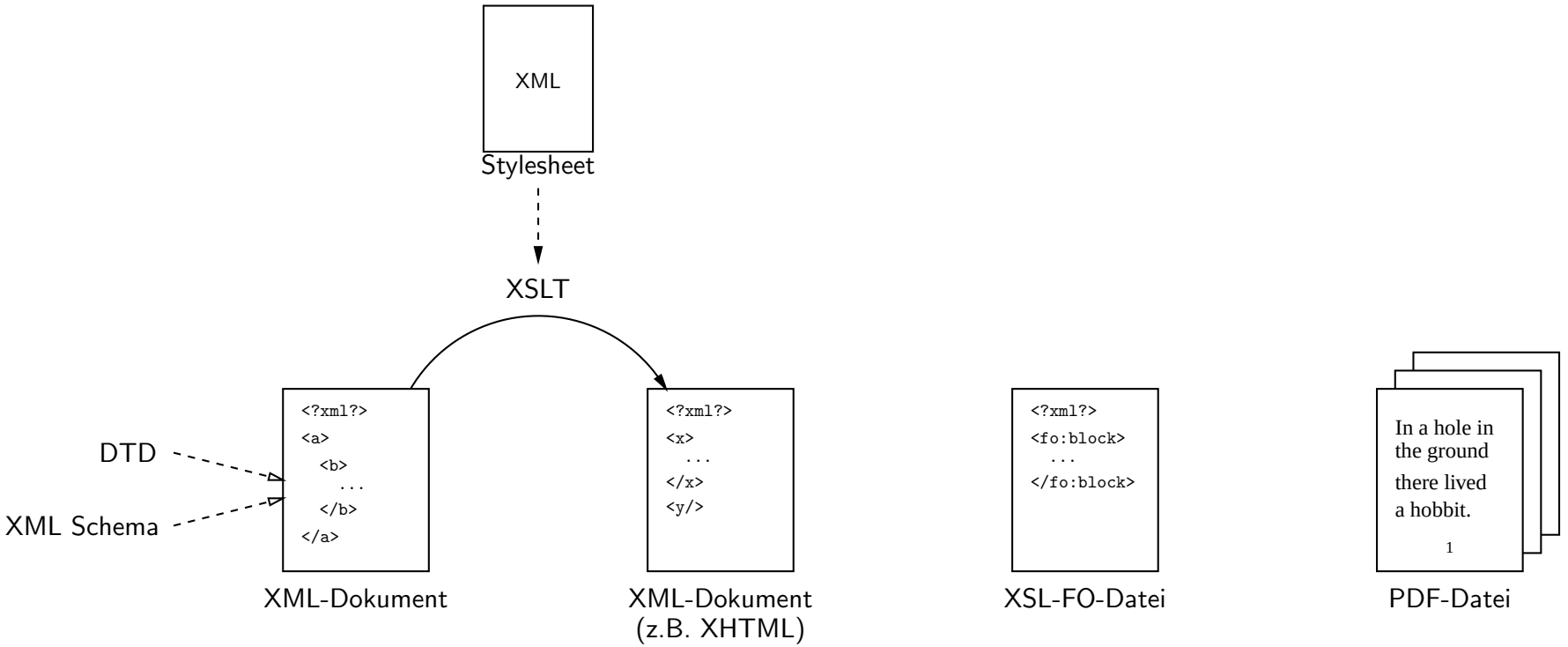

DTD: Document Type Definition
XSL: Extensible Style Sheets
XSLT: XSL Transformations

# Verarbeitung von XML




DTD: Document Type Definition
XSL: Extensible Style Sheets
XSLT: XSL Transformations

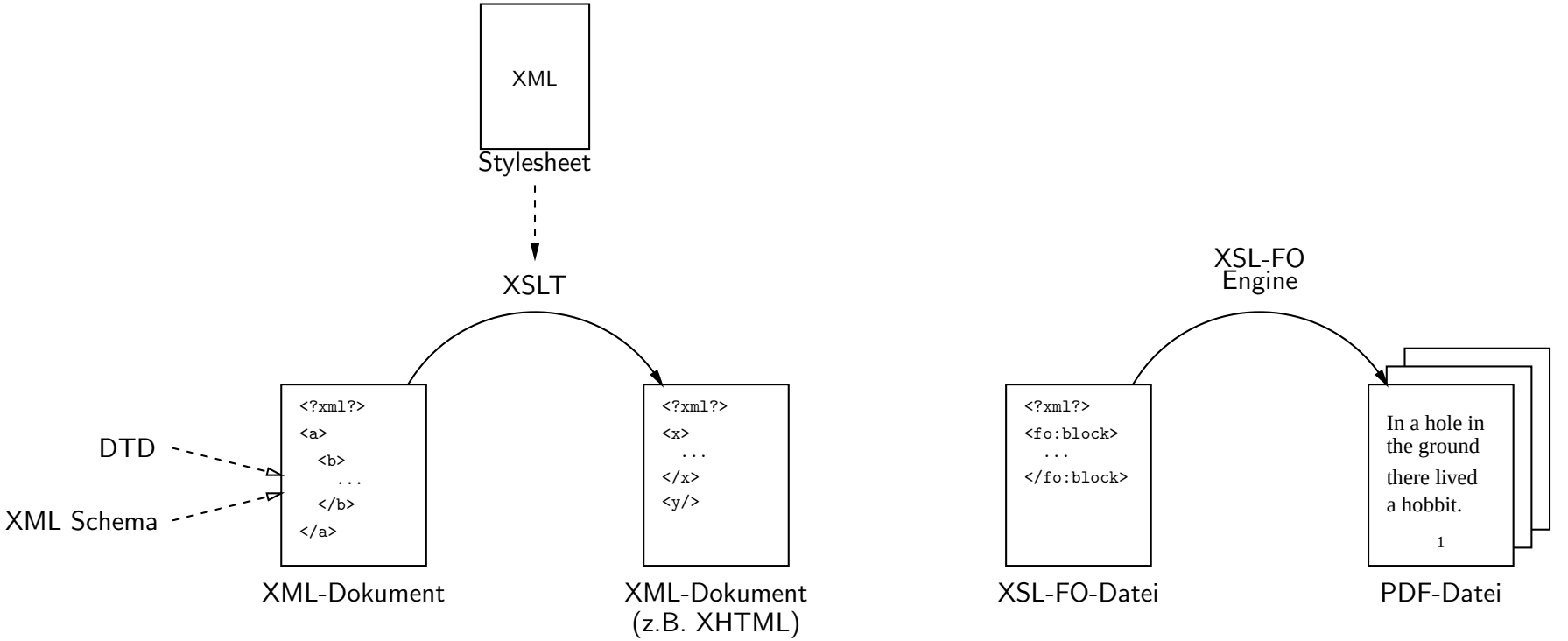
# Verarbeitung von XML




DTD: Document Type Definition
XSL: Extensible Style Sheets
XSLT: XSL Transformations

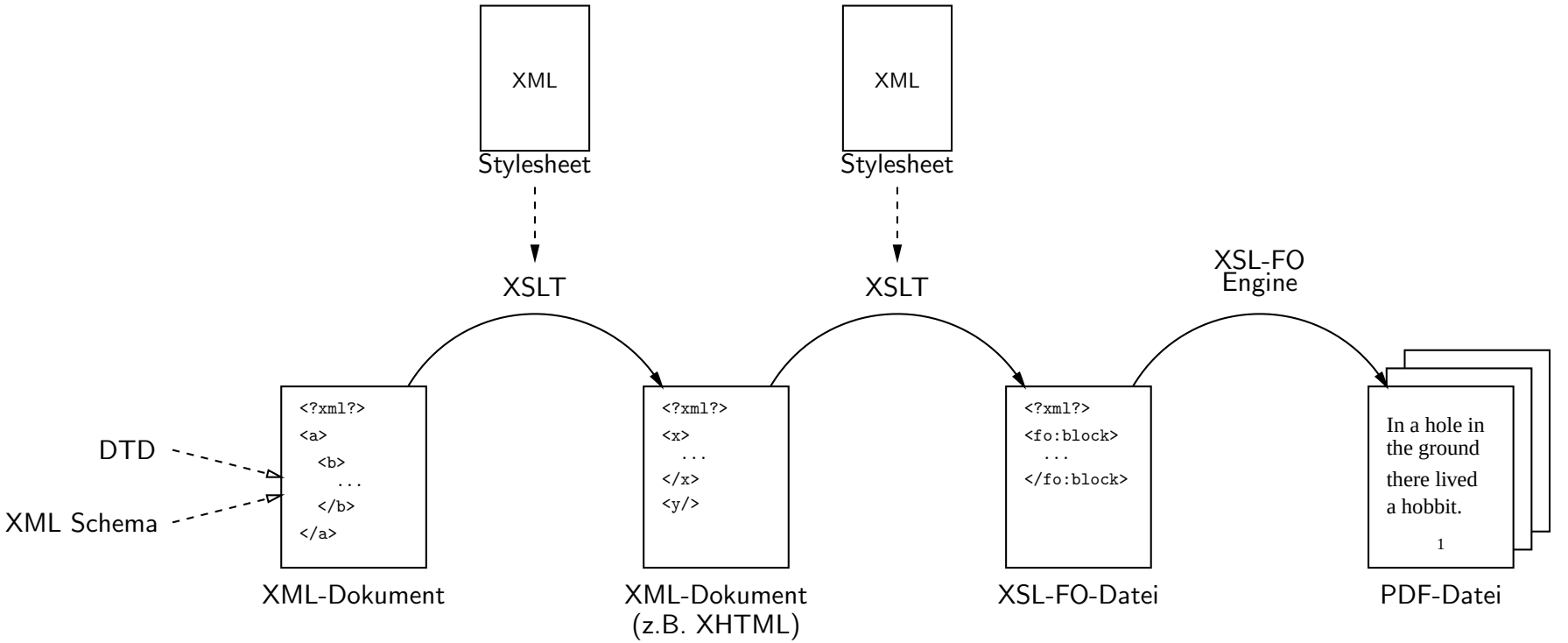
FO: Formatting Objects

# Verarbeitung von XML



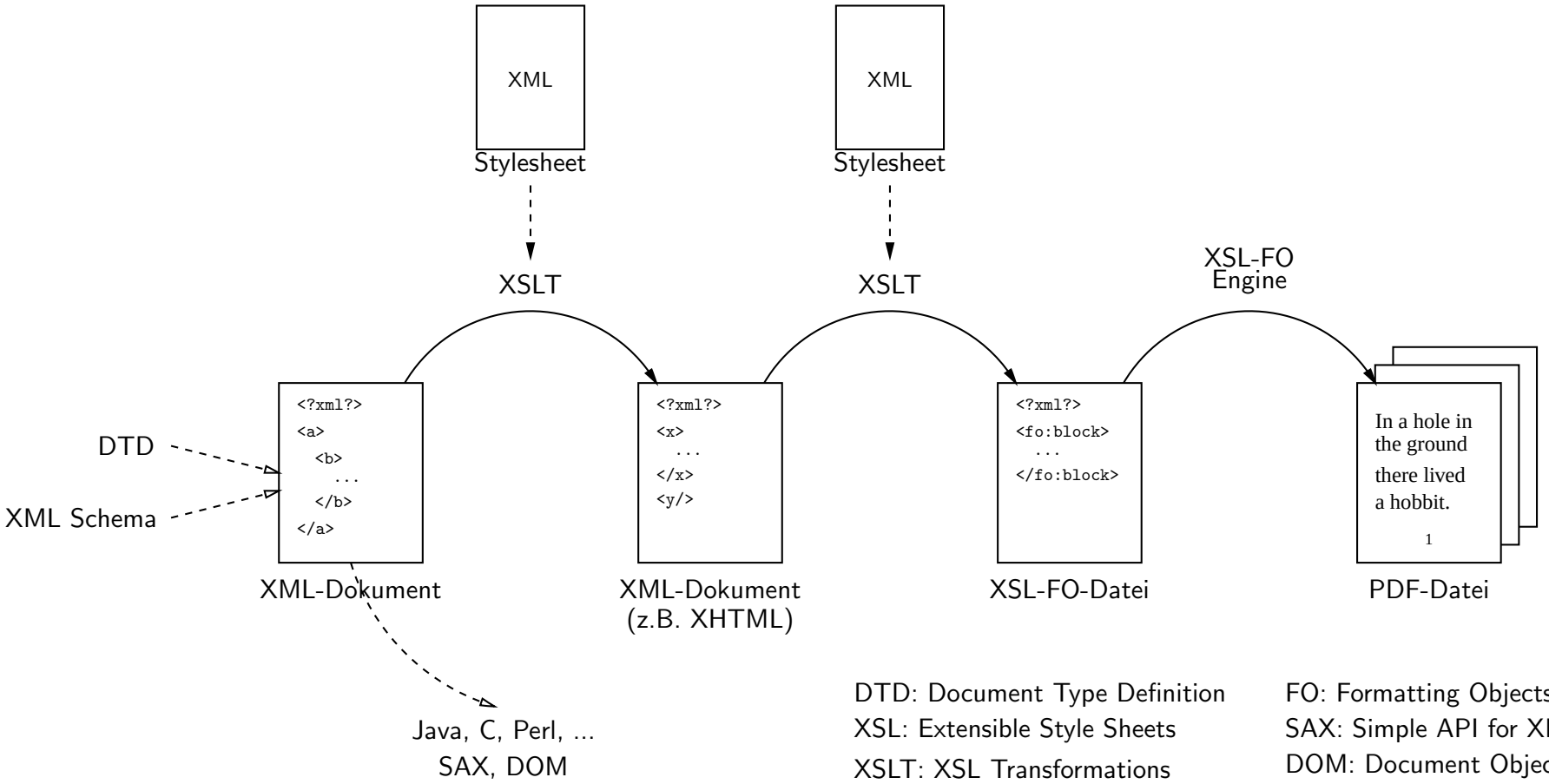

DTD: Document Type Definition
XSL: Extensible Style Sheets
XSLT: XSL Transformations
FO: Formatting Objects

# Verarbeitung von XML




DTD: Document Type Definition
XSL: Extensible Style Sheets
XSLT: XSL Transformations
FO: Formatting Objects

# Verarbeitung von XML




DTD: Document Type Definition
XSL: Extensible Style Sheets
XSLT: XSL Transformations

FO: Formatting Objects
SAX: Simple API for XML
DOM: Document Object Model

# Verarbeitung von XML, cont.

- (fast) alles ist XML
- wenige Werkzeuge nötig (XML-Parser, XML-Editor, . . . )
- wiederverwendbare Komponenten
- Textformat Unicode: portabel, einfach zu verarbeiten
- offene Standards, viele Open Source-Lösungen

# ein Beispiel

```
<?xml version="1.0"?>
<presentation status="draft" date="2002-10-04">
  <title>XML &amp; Friends for Dummies</title>
  <author>Joe User</author>
  <slide>
    <title toc="yes">What is XML?</title>
    <ilist>
      <item>XML is not a markup language (unlike HTML)</item>
      <item>XML instances can be <emph>well formed</emph> or
        even <emph>validating</emph></item>
      <item>XML stands for &xml;</item>
    </ilist>
  </slide>
  <slide>...</slide>
</presentation>
```

# Aufbau von XML

- XML-Datei beginnt mit *XML declaration*

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

- bisher nur Version 1.0 spezifiziert, neue Versionen in Planung
- verwendete Kodierung

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

- sonst Unicode (UTF-8), Obermenge von ASCII
- Kodierung im Dokument nicht unproblematisch

# Elemente (elements)

- öffnendes und schließendes *tag*

```
<item>XML is not a ...</item>
```

- Elemente können geschachtelt werden

```
<ilist>
  <item>XML is not a ...</item>
  <item>... <emph>well formed</emph> ...</item>
</ilist>
```

- keine Minimierungsregeln
- leeres Element

  `<hr/>` statt `<hr></hr>` statt `<hr>`

# Elemente, cont.

- Schachtelung muß „passen“

```
<a> <b> ... </a> </b>
```

- Groß/Klein-Schreibung relevant

```
<html> ... </HTML>
```

- XML-Dokument muß genau ein oberstes Element enthalten

```
<?xml version="1.0"?>
<presentation>
  ...
</presentation>
```

# Attribute (attributes)

- Zusatzinformationen zu Elementen

```
<presentation status="draft" date="2002-10-04">
```

- nur im öffnenden tag
- Anführungszeichen " (double quote) oder ’ (single quote)
- Attribut darf nur einmal vorkommen

```
<presentation lang="en" lang="de">
```

- Designfrage: Wann Elemente, wann Attribute?

```
<date y="2002" m="10" d="7"/>
<date><y>2002</y><m>10</m><d>7</d></date>
```

# Entitäten (entities)

- Makros und Sonderzeichen
- in XML vordefinierte *entity references*

  `& < > &apos; &quot;`
- weitere können definiert werden
- *character references*: Zugriff auf beliebige Unicode-Zeichen

  `&#xA9; 2002 by J&#xF6;rn Clausen`

# wohlgeformtes vs. valides XML

- *well formed XML*: bestimmte Kriterien erfüllt
  - korrekte Kodierung
  - korrekte Schachtelung der Elemente
  - korrekte Attribute
  - definierte Entitäten
- XML-Parser kann Wohlgeformtheit überprüfen
- *valid XML*: Dokument entspricht einer Grammatik

# XML-Grammatiken

- formale Beschreibung der Syntax eines XML-Dokuments
  - Welche Elemente gibt es?
  - Wie dürfen/müssen Elemente geschachtelt werden?
  - Welche Attribute dürfen in welchen Elementen vorkommen?
  - Welche Entitäten sind definiert?
- erleichtert den Austausch von XML-Dokumenten
- kann bei der Erstellung eines XML-Dokuments helfen

# XML-Grammatiken, cont.

- von SGML übernommen: *Document Type Definition* (DTD)
  - für XML etwas eingeschränkt
  - DTD selber ist nicht XML
- alternative Vorschläge:
  - W3C XML Schema
  - RelaxNG
  - Schematron
  - Document Schema Definition Language (ISO/IEC 19757)
- zur Zeit *sehr* kontrovers diskutiert

# Document Type Definition

- für allgemeine Daten nicht immer ideal
- für textuelle Daten aber (immer noch) geeignet
- große Anzahl an etablierten DTDs
- große Anzahl an Werkzeugen (XML/SGML-Editoren)

# Elemente definieren

- Element besitzt *content model*

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

- #PCDATA: Text ohne weitere Elemente
- Container-Elemente

```
<!ELEMENT  slide          (title, ilist)>
<!ELEMENT  ilist          (item*)>
```

- Quantoren: ?, +, * (wie in regulären Ausdrücken)
- Konnektoren:
  - a, b: erst a, dann b
  - a | b: a oder b

# Elemente definieren, cont.

- Mischung von Text und *inline*-Elementen

```
<item>... can be <emph>well formed</emph>
      or even ...</item>
```

- *mixed content*

```
<!ELEMENT  item          (#PCDATA | emph)*>
```

- kann bei der Verarbeitung problematisch sein

# Attribute definieren

- „Aufzählungstyp“

```
<!ATTLIST  presentation
           status  (draft|final|publ)   #REQUIRED>
```

- Vorgabe definieren

```
<!ATTLIST  presentation
           status  (draft|final|publ)   "draft">
```

- beliebiger Text

```
<!ATTLIST  presentation
           status  (draft|final|publ)   #REQUIRED
           date    CDATA                #IMPLIED>
```

# Entitäten definieren

- textuelle Makros

```
<!ENTITY  xml  "Extensible Markup Language">
```

- Umlaute wie in HTML

```
<!ENTITY   Auml   "&#196;">
<!ENTITY   auml   "&#228;">
```

- *parameter entities*, für Makros in der DTD

```
<!ENTITY   % text "(#PCDATA|ital|bold)*">
<!ELEMENT  item   %text;>
<!ELEMENT  ital   %text;>
<!ELEMENT  bold   %text;>
```

# DTD einbinden

- Verweis auf DTD im XML-Dokument
- *system identifier*

```
<?xml version="1.0"?>
<!DOCTYPE presentation SYSTEM "presentation.dtd">
```

- *public identifier* und system identifier

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```