Web-Programmierung (WPR)

Vorlesung II.

XML

Manfred Gruner

mailto:wpr@gruner.org

4.2 XML

- XML
  eXtensible Markup Language
- Universelles Format für strukturierte Dokumente und Daten
- Web: XML = Querschnittstechnologie
- Anwendungsszenarien
  - Konfigurationsdateien
  - Austauschformat zwischen WebClient u. Server

4.2 XML

- Vorteile von XML
  - Plattformneutral
  - Trennung von Inhalt und Präsentation
  - Repräsentation beliebig komplexer strukturierter Daten
  - Anreicherung von Nutzdaten um Metainformationen
  - Verbesserung der Qualität von Dokumenten- und Datenverarbeitungsprozessen (DTD,..)
  - Verfügbarkeit standardisierter Tools

4.2 XML

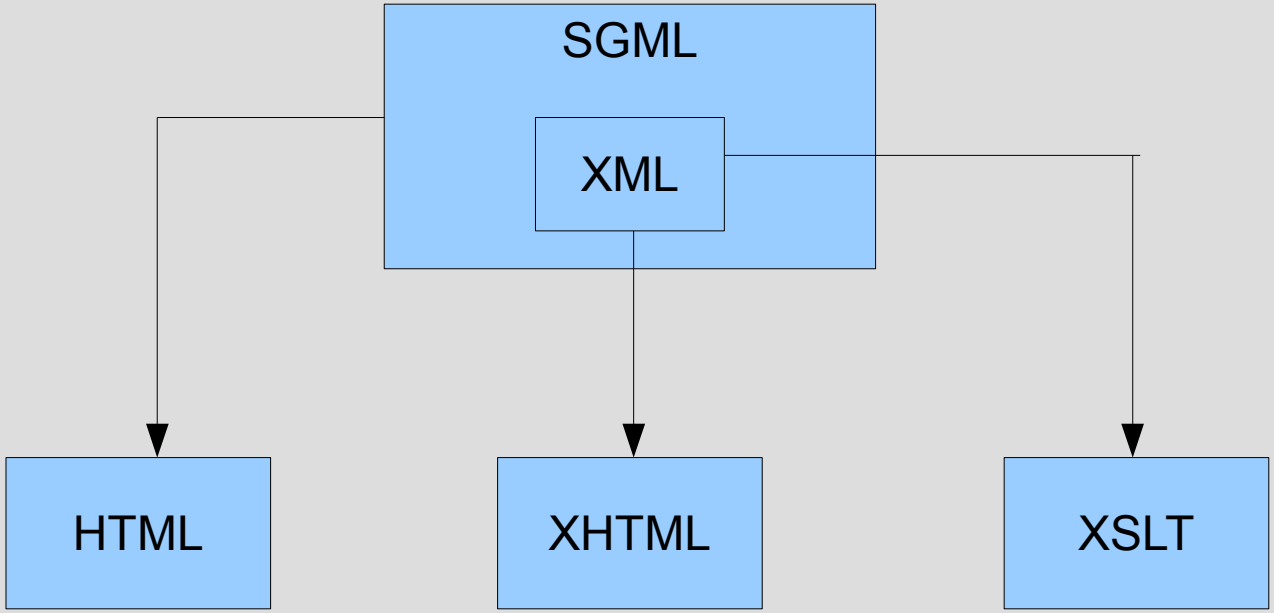
- XML-Co-Standards
  - XSL
    eXtensible Stylesheet Language
    (Formatierung und Layout)
  - XSLT
    XSL Transformation
  - XPath
    XML Path Language
    (Zugriff auf Dokumentenknoten)
  - XML-Schema
    Dokumententypen-Definition

4.2.1 Markup-Sprachen

- Markup = Auszeichnung von Text
- Wesentlichen Vorteile
  - Grundsätzliche plattformneutral
  - Lesbar durch Mensch u. Computer
  - Veränderbar durch variable Länge
- Formatierendes Markup (HTML)
  - Schriftart + Schriftgröße + ...
- Generisches Markup
  beschreibt Semantik des ausgez. Textes
  Tagnamen = Metainformationen

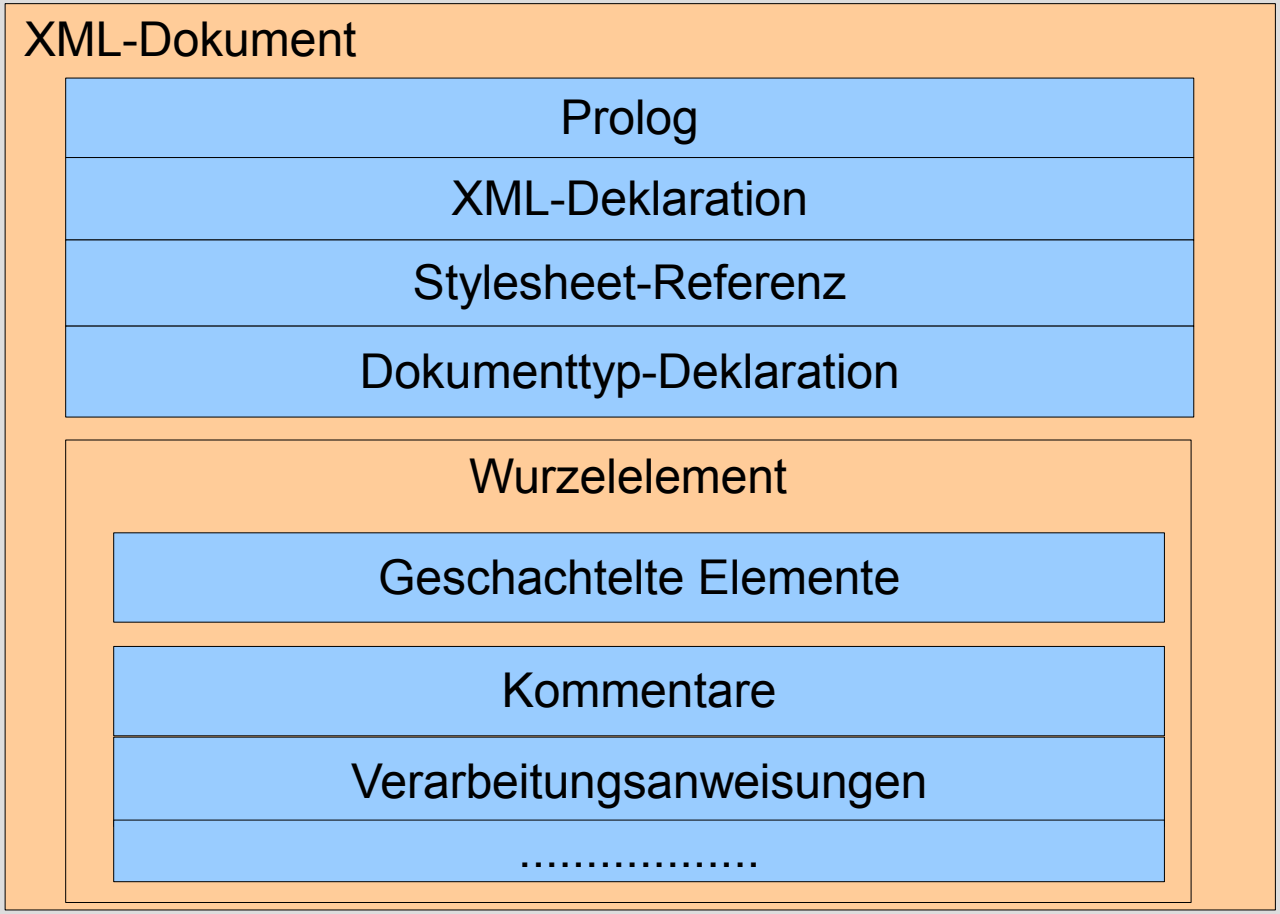
4.2.1 Markup-Sprachen

- Erweiterbares Markup
  - HTML = fester Umfang von Tags
  - XML = erweiterbar



4.2.2 XML-Dokument

- Struktur



Beispiel

4.2.2 XML-Dokument

- Formale Auszeichnungsregeln
  - Ein XML-Dokument hat genau ein Wurzelelement
  - XML-Elemente müssen mit einem Ende Tag abgeschlossen werden
  - XML-Element müssen korrekt geschachtelt sein
  - Attributwerte müssen in Anführungszeichen oder Hochkomma stehen
  - Attribute müssen als Name=Wert-Paar definiert werden

```
<td nowrap>Inhalt</td>
```

4.2.2 XML-Dokument

- XML Dokument
  - Elementen
  - Attributen
  - Textuellen Inhalt
- Elemente
  - mit Inhalt (Text oder andere Elemente)
  - ohne Inhalt
  - Gemischer Inhalt (Text und andere Elemente) !
- Jedes Element muss genau ein Parent-Element haben

4.2.2 XML-Dokument

- Attribute
  - Beschreiben ein Element zusätzlich
  - Kardinalität:1
- Modellierungsproblematik
  Attribut oder Element ??
  - Eigentliche „Bestimmung“ von Attributen:Metadaten
  - Child-Elemente enthalten eigentliche Informationen

4.2.2 XML-Dokument

- Regeln für XML-Namen
  - Unterscheidung Groß- u. Kleinschreibung
  - Namen beginnen mit Buchstaben oder Unterstrich (Mindestlänge 1)
  - Ziffern, Groß- Kleinbuchstaben, Unterstrich, Bindestrich(-), Punkt(.)
    Zeichen aus anderen Sprachräumen erlaubt
  - Namen dürfen „xml“ nicht enthalten

4.2.2 XML-Dokument

- Entity Referenzen
  Grund:
  Verbot von manchen Zeichen im textuellen Inhalt
  < , > , & , “ , '
  Ersatzdarstellungen
- XML besitzt 5 vordefinierte Entity-Referenzen

Zeichen	Entity-Referenz
<	&lt;
>	&gt;
&	&amp;
„	&quot;
'	&apos;

4.2.2 XML-Dokument

- CDATA Bereich
  character data

  ```
  <![CDATA[ nicht zur parsender Text]]>
  ```

  Bereich wird vom Parser nicht interpretiert

- Zeichenreferenzen (Unicode)

Zeichen	dezimal	hexadezimal
ö	&#246;	&#xF6;

4.2.3 Dokument-Typen

- Dokument
  - Inhalt, Präsentation,Struktur

    Dokumentenstruktur legt Elemente fest aus denen sich ein Dokument zusammensetzen kann.

- Dokumententyp beschreibt Dokumente von ein und derselben Struktur
  - HTML 4.0, XMHTML

- XML-Dokumenttypen = XML-Anwendungen

4.2.3 Dokument-Typen

- Status eines XML-Dokument
  - Well formed
    Dokument entspricht formalen Auszeichnungsregeln
  - Valid
    Struktur des Dokumentes entspricht den Regeln des zugehörigen Dokumententyps.

    Wichtig:
    Valid = Well Formed
    Well Formed != Valid

- Dokumenttypen => Verbesserung der Qualität

4.2.3 Dokument-Typen

- Dokumenttyp-Definition
  - DTD
    - DTDs besitzen keine XML-Syntax
    - Elemente und Attribute werden definiert

      Element-Deklaration definiert Element und legt Regeln für dessen Verwendung fest.

      <ELEMENT Name Inhaltsmodell>

      Attributlistendeklaration

      <!ATTLIST Elementname Attributname Attributtyp Voreinstellung>

4.2.3 Dokument-Typen

- Dokumenttyp-Definition
  - Definition Reihenfolge („ , “)
  - Definition Optionen („ | “)
  - Definition Kardinalität
    - ? => 0 ... 1 (= optional)
    - + => 1 ... n
    - * => 0 ... n

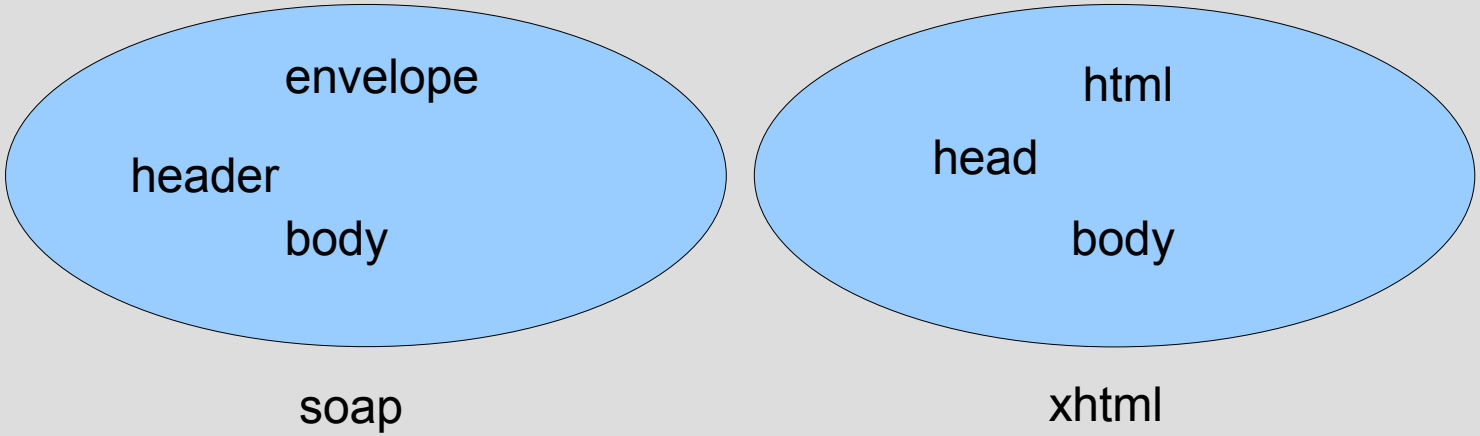
      wird Elementnamen nachgestellt
  - Inhaltsmodell EMPTY,ANY

Siehe Beispiel wein.dtd

4.2.4 Namesräume

- Ziel: Vermeidung von Namenskonflikten

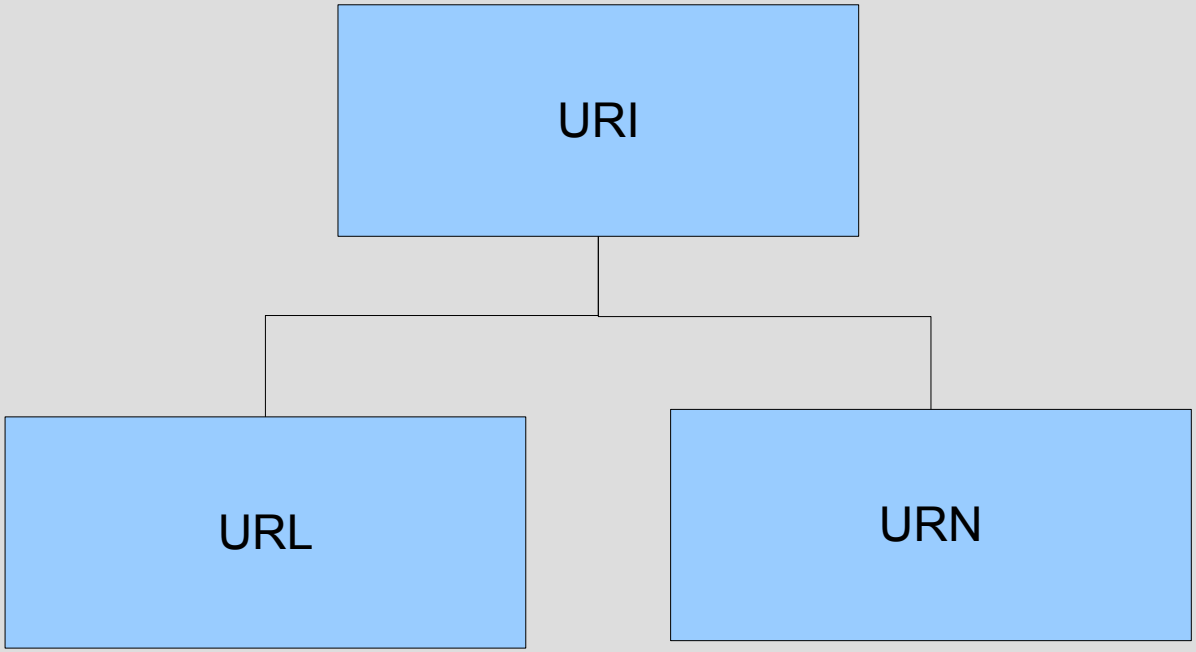
  XML-Namensraum (namespace)
  - Kollektion von Element- und Attributenamen



Elemente einer XML-Applikation werden Präfixe vorangestellt: `soap:body`

4.2.4 Namesräume

- Namensraum-Deklaration



Namensräume verknüpfen Elemente mit URIs

`<element xmlns:`*`präfix`*`=URI>`

4.2.4 Namesräume

- Problem der Eindeutigkeit gibt es auch für Präfixe
- Präfixe müssen nur in einem Dokument eindeutig sein
- Der benutzte URI muss nicht wirklich existieren.
- Namesraumdeklaration im Start-Tag des Wurzelknoten
- Child-Tags erben den Nameraum
- DTDs unterstützen keine Namensräume. Beispiel: wein3.xml

4.2.5 Dokumente u. Daten

- XML war gedacht um Dokumente zu spezifizieren
- Heutzutage XML=universelles Datenaustauschformat
  - Hierarchischer Struktur
  - Plattformneutralität
- Aus Daten werden XML-Dokumente
- Datenaustauschformat erfordert weitergehende Prüfungen.

4.2.5 Dokumente u. Daten

- XML-Schema
  Alternative zu DTDs

- Vorteile von XML-Schema
  - Ein XML-Schema ist ein XML-Dokument
  - Präziesere Inhaltsmodellbeschreibung möglich
  - Alle aus anderen Programmiersprachen bekannten Typen stehen zur Verfügung (für Elemente und Attribute)
  - Namensräume werden unterstützt
  - Ein XML-Dokument kann mit mehreren Schemas verknüpft werden.

4.2.5 Dokumente u. Daten

- Element-Deklaration

```
<xsd:element name="name"
             type="xsd:string"/>
```

- Einfach Elemente (Blätter der Struktur)
  - enthält keine Elemente und Attribute
  - Verbindet Namen mit Typ
  - Mögliche Typen:
    boolean, byte, integer, float, double, duration, dateTime
- Eigene Typen und Wertebereiche definierbar

4.2.5 Dokumente u. Daten

- Komplexe Elemente

```
<xsd:element name="name">
   <xsd:complexType>
      ....
   </xsd:complexType>
</xsd:element>
```

- Eigene Typedefinition möglich

Beispiel: wein4.xml

4.2.6 XML-Verarbeitung

- 1. Schritt bei der Verarbeitung von XML
  ==> parsen
  2 Programmiermodelle
  - SAX (Simple API for XML)
  - DOM (Document Object Model)

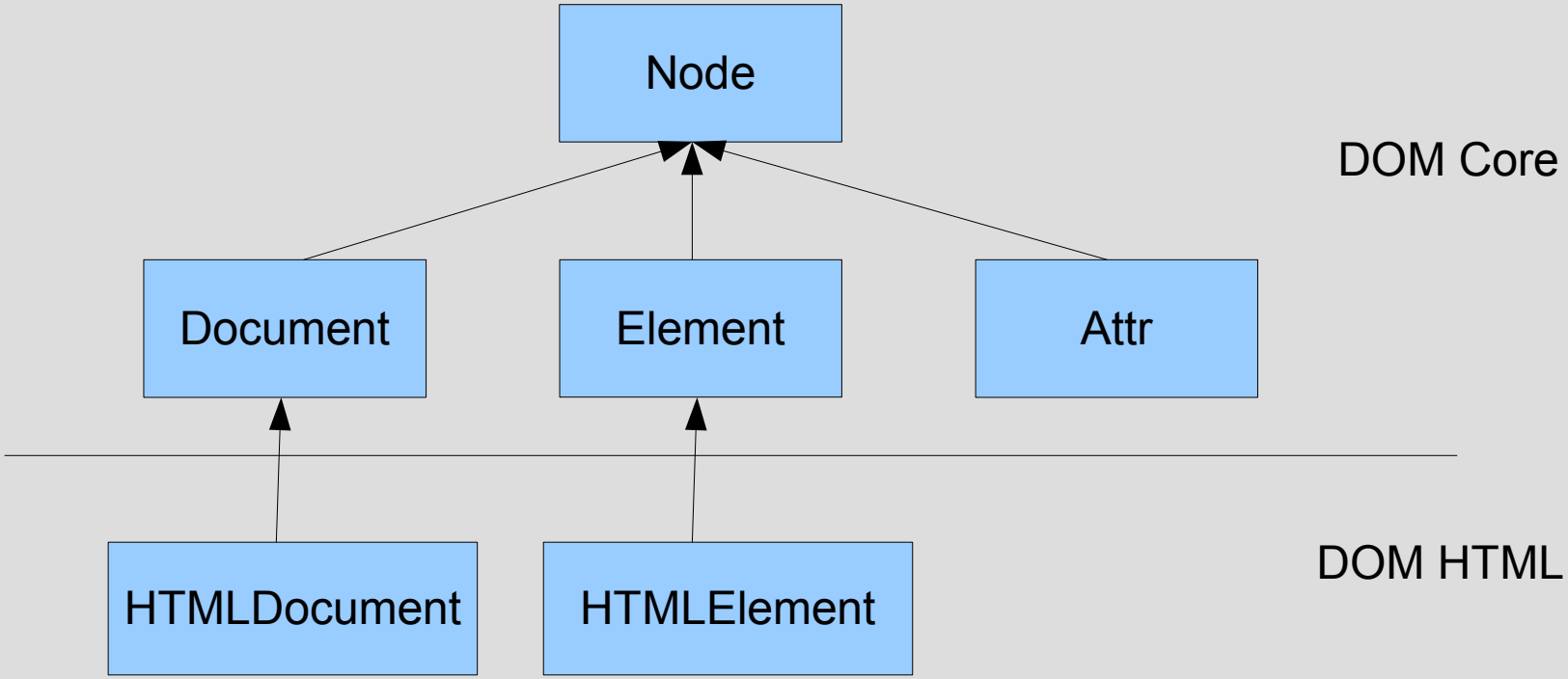
4.2.6 XML-Verarbeitung

- SAX (Simple API for XML)
  - XML-Dokument verfügbarmachen über Callbacks
  - ContentHandler-Klasse muss implementiert werden
  - Events werden generiert
  - Verarbeitung und parsen geschieht quase gleichzeitig

	SAX-Parser Bild

4.3 DOM

- Document Object Model



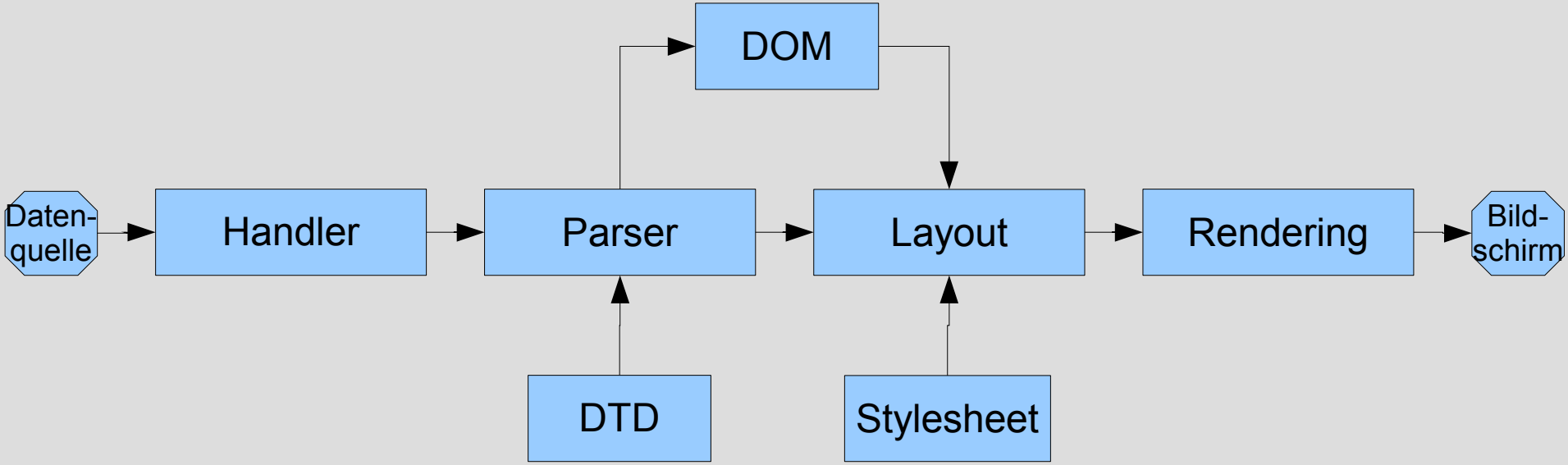
DOM in Firefox

4.3 DOM

- Wichtigster Unterschied zu SAX
  - DOM Parser
    - Langsam (zuerst wird ganze Dokument geparst und DOM aufgebaut im Speicher)
    - Danach erst Zugriff auf DOM möglich
    - Manipulierung des DOM durch DOM-API möglich (u.a. AJAX)
  - Implementierung für unterschiedliche Programmiersprachen (C,C++,**Java, JavaScript**, ...)

4.4 Layout-Prozess

Vereinfachte Darstellung des Layoutprozesses für HTML-Dokumente



4.5 Ereignisorientierung