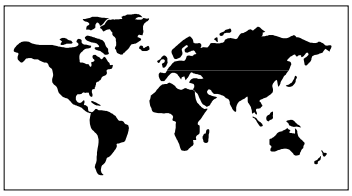
WWW – Grundlagen und Technologie

Extensible Markup Language (XML)



Erik Wilde
TIK – ETH Zürich
Sommersemester 2001

Übersicht

- Motivation für die Einführung von XML
- XML Positionierung
  - ein *Profile* von SGML
- XML Basics
  - Szenarien
  - Syntax
  - DTDs
- XML Schema
  - Weiterentwicklung von DTDs
- Zusammenfassung

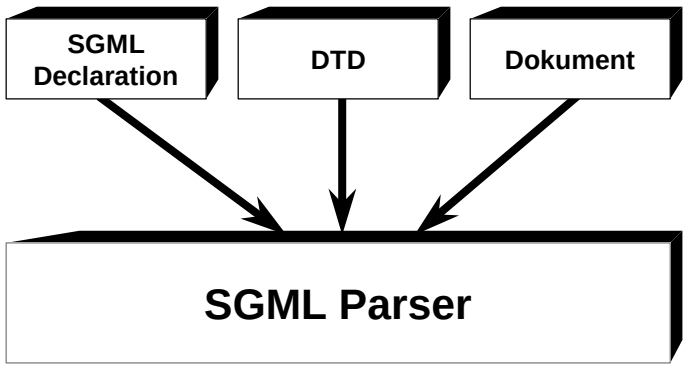
Probleme mit HTML

- HTML als Standard des “prä-XML Web”
- HTML ist präsentationsorientiert
  - erste Probleme schon seit Beginn des “Browser War”
  - trotz Verbesserungen (CSS) bleibt der Grundansatz
  - HTML basiert auf reiner Präsentationssemantik
- gewünscht: Wiederverwendbarkeit von Inhalten
  - HTML als Dokumentenformat ungeeignet
  - HTML nur eine mögliche Form der Präsentation
- Content Mangement wird immer wichtiger
  - fehlende Freiheit, eigene Strukturen zu verwenden
  - fehlende Flexibilität, auf Neues zu reagieren (WML)

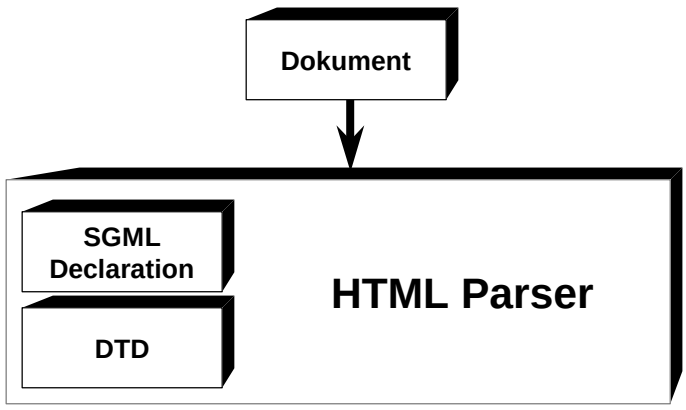
XML und HTML

- sieht “so ungefähr” aus wie HTML
  - gleiche Grundlage (SGML)
  - *proven success* (SGML und HTML sind Erfolge)
  - geringere Hemmschwelle für Umsteiger
- funktioniert ähnlich wie HTML
  - gleiche Strukturierungsverfahren (Grammatiken)
  - rein textorientiertes Format (keine Binärdaten!)
- andere Zielgruppe als HTML
  - weiterverarbeitbare Information (B2B)
  - anwendungsabhängige Datenstrukturen
  - etabliertes Umfeld (EDI, SGML, proprietär)

SGML Parser



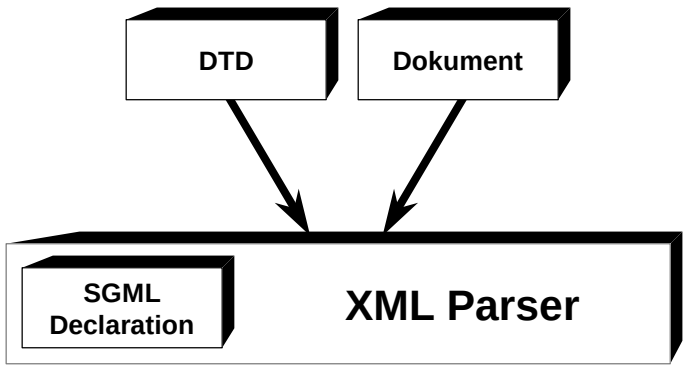
HTML Parser



Extensible Markup Language (XML)

- benutzerdefinierbare Dokumenttypen
- überwindet Einschränkungen von HTML
  - beliebige Dokumenttypen
  - neues Problem: Semantik von Elementen
  - begleitende Mechanismen werden notwendig
- überwindet Komplexität von SGML
  - fest definierte konkrete Syntax (SGML Declaration)
  - keine Markup Minimization (immer volles Markup)
  - reduzierte Zahl an erlaubten Attributtypen
- Ziele sind Einfachheit und Flexibilität

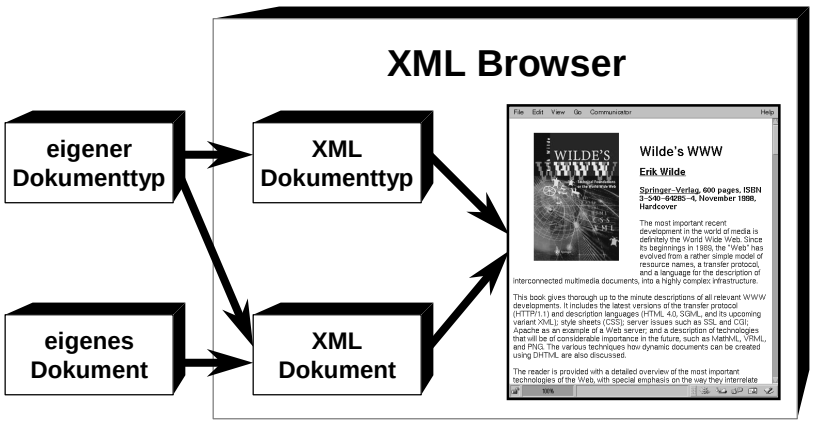
XML Parser



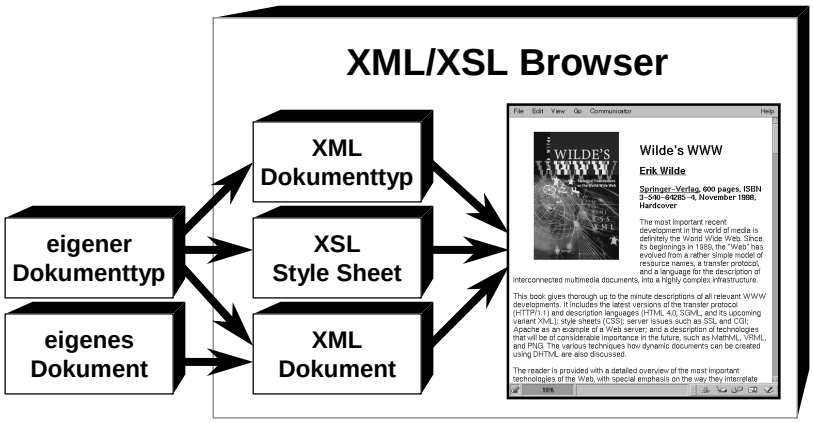
Vergleich SGML/XML/HTML

	SGML	XML	HTML
SGML Declaration	frei	fix	fix
DTD	frei	frei	fix
Dokument	frei	frei	frei

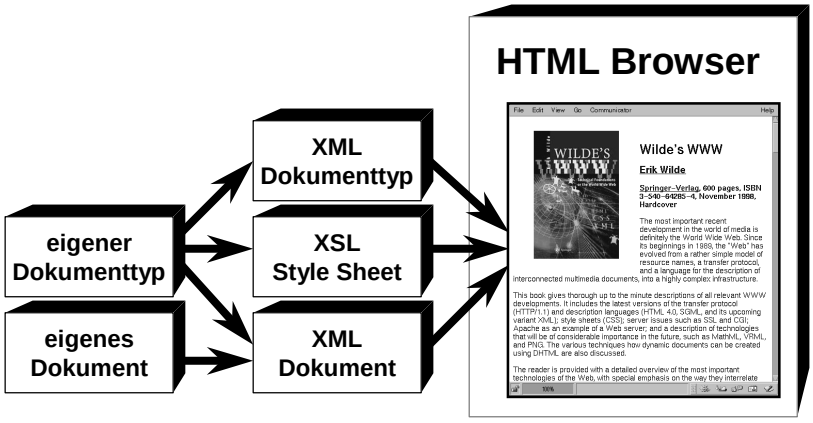
Publishing mit XML



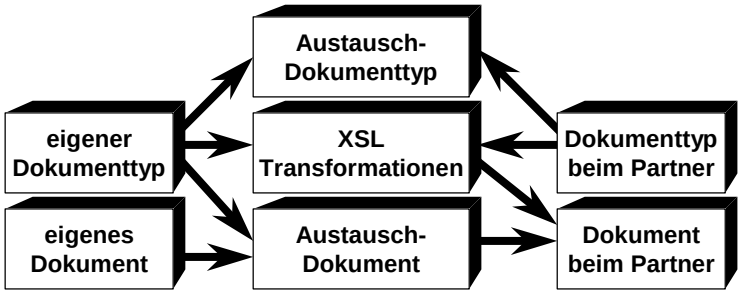
Publishing mit XML (Client-Side)



Publishing mit XML (Server-Side)



XML (Server-Side) als B2B-Lösung



XML unabhängig vom WWW

“XML is the silver bullet of electronic commerce”

- XML als Datenformat in B2B-Anwendungen
  - Anforderungen sind eher bescheiden (funktional)
  - Problem der Einigung auf Standards (EDI, ASN.1)
- was ist neu an XML in B2B Szenarien?
  - Erweiterbarkeit und Robustheit
  - geringe *Barrier to entry* (Kosten, Komplexität)
  - breite Akzeptanz in verschiedensten Bereichen

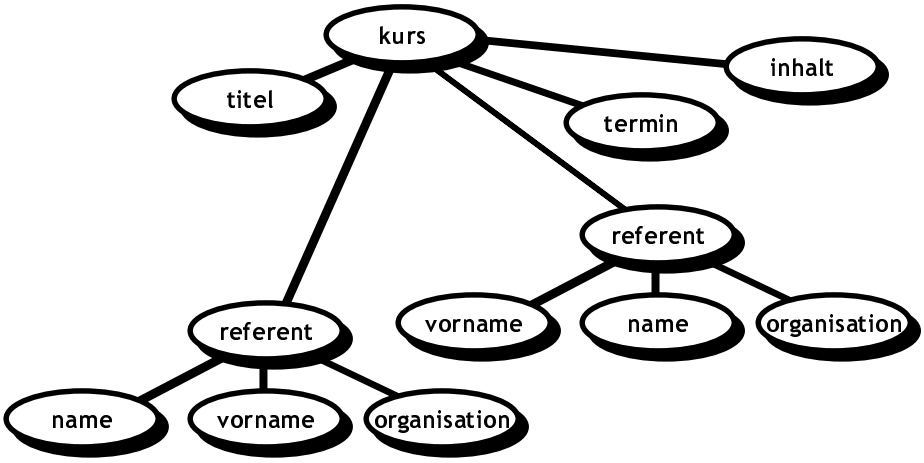
Beispiel (XML)

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

<kurs>
<titel kurz="XML">XML - Grundlagen und Umfeld</titel>

<referent email="xml@dret.net"
          homepage="http://dret.net/">
  <vorname>Erik</vorname>
  <name>Wilde</name>
  <organisation homepage="http://www.tik.ee.ethz.ch/">ETH
  Zürich</organisation>
</referent>

<referent> ... </referent>

<termin date="20000512" location="technopark"/>

<inhalt> ... </inhalt> </kurs>
```


Markup

- Markup ist die physische Form eines Dokuments
  - Markup ist immer menschenlesbar
  - *XML Text = Character Data + Markup*
- Markup wird durch spezielle Zeichen markiert
  - Tags sind in `<` und `>` eingeschlossen (`<titel>`)
  - Entities sind in `&` und `;` eingeschlossen (`&#252;`)
- XML benutzt immer die gleichen Zeichen
  - wichtiger Unterschied zu SGML (*SGML Declaration*)
  - ermöglicht einfachere Implementierungen
- Markup-Analyse ist eine Standardaufgabe
  - eingebaut in Software (z.B. MSXML in IE)

Baumstruktur von XML



Elemente

- Elemente sind der grundlegende Mechanismus
  - Strukturierung von hierarchischen Daten
  - "beliebige" Namensgebung für Elemente
  - Definition gemäss inhaltlichen Strukturen
  - Kernpunkt des DTD-Designs
- Elementtypen haben zwei wichtige Aspekte
  - ein *content model* für erlaubten Inhalt
  - Attribute (optionales Vorkommen oder notwendig)
- DTD deklariert Typ, den Dokument verwendet

```
DTD:       <!ELEMENT titel      (#PCDATA) >
Dokument: <titel>XML - Grundlagen und Umfeld</titel>
```

Regeln für Elementdeklarationen

- der Inhalt von Elementen kann sein
  - nur Elemente *(element content model)*
  - Text gemischt mit Elementen *(mixed content model)*
  - kein Inhalt erlaubt (**`EMPTY`** Keyword)
- das *content model* eines Elements
  - optionales Vorkommen mit **`?`**, wiederholbar mit **`*`**
  - notwendig und wiederholbar mit **`+`**
  - Sequenz mit **`,`**
  - Alternativen (Exklusiv-oder) mit `|`
- nicht erlaubt sind folgende SGML Konstrukte
  - vertauschbare Vorkommen mit **`&`** und Exceptions

Regeln für Elemente

- jedes XML-Dokument hat genau eine Wurzel
  - *document element*
- jedes Element hat ein Eltern-Element
  - das *document element* hat kein *parent element*
- direkt untergeordnete Elemente sind Kinder
  - falls keine Kinder: Blätter (*leave element*)
- untergeordnete Elemente sind Nachkommen
  - *descendant* (Kinder und Kindeskinder usw.)
- übergeordnete Elemente sind Vorfahren
  - *ancestor* (Eltern und Grosseltern usw.)

Attribute

- Attribute sind Informationen zu Elementen
  - Attribute geben Zusatzinformationen
  - Entscheidung Attribut/Element nicht immer klar
- optional (`#IMPLIED`) oder notwendig (`#REQUIRED`)
- Attribute können verschiedene Typen haben
  - ein Konzept, das für Elemente nicht existiert
  - deutliche Einschränkungen (siehe HTML DTD)

```
DTD:       <!ATTLIST titel kurz CDATA #REQUIRED >
Dokument: <titel kurz="XML">XML - Grundlagen...
```


Regeln für Attributdeklarationen

- erlaubt sind
  - mehrere Attributtypen in einer Attributliste
  - mehrere Attributlisten für ein Element
  - bei Namenskonflikten zählt das erste Vorkommen
  - gleiche Attributnamen für verschiedene Elemente
- nicht erlaubt sind
  - eine Attributliste für mehrere Elemente (erlaubt in SGML!)
- erlaubte Typen sind
  - String types (beliebiger String als Wert)
  - Tokenized types (XML Namen verschiedener Art)
    - insbesondere `ID`/`IDREF(S)` als Referenzierungsmechanismus
  - Enumerated types (Auswahl aus definierter Liste)

Regeln für Attribute

- ein Attribut ist immer ein Name/Value-Paar
- Attributnamen müssen also angegeben werden
  - in SGML/HTML dürfen sie u.U. weggelassen werden
- Attributwerte müssen in Quotes gesetzt werden
- Attribute können weggelassen werden
  - vom Parser ersetzt falls auf **`#IMPLIED`** gesetzt
  - nicht erlaubt falls auf **`#REQUIRED`** gesetzt
- Attribute werden immer im Start-Tag verwendet
  - konzeptionell Information am Element-Knoten

Document Type Definition (DTD)

- Beschreibung der Datenstrukturen in einem Schema
  - Schema beschreibt eine Klasse von Dokumenten
  - SGML/XML DTD ist nur eine mögliche Variante
  - *XML Schema* als Weiterentwicklung (später mehr dazu...)
- Beschreibung von Datenblöcken
  - Elemente als Strukturmittel
  - Attribute als Daten zu Elementen
- Beschreibung der erlaubten Kombinationen
  - Definition einer Grammatik
  - Verwendung für die Validierung von Daten
  - Verwendung für die Generierung von Daten
- Schema Modellierung als Kern von XML

Beispiel (Verweis auf DTD)

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

<kurs>
<titel kurz="XML">XML - Grundlagen
  und Umfeld</titel>

<referent email="xml@dret.net"
  homepage="http://dret.net/">
   <vorname>Erik</vorname>
   <name>Wilde</name>
```


Beispiel (Teil einer DTD)

```
<!ELEMENT kurs         (titel, referent+, termin+, inhalt) >

<!ELEMENT titel        (#PCDATA) >
<!ATTLIST titel
          kurz         CDATA #REQUIRED >

<!ELEMENT referent     (vorname, name, organisation?) >
<!ATTLIST referent
          email        CDATA #IMPLIED
          homepage     CDATA #IMPLIED >

<!ELEMENT vorname      (#PCDATA) >

<!ELEMENT name         (#PCDATA) >

<!ELEMENT organisation (#PCDATA) >
<!ATTLIST organisation
          homepage     CDATA #IMPLIED >
```

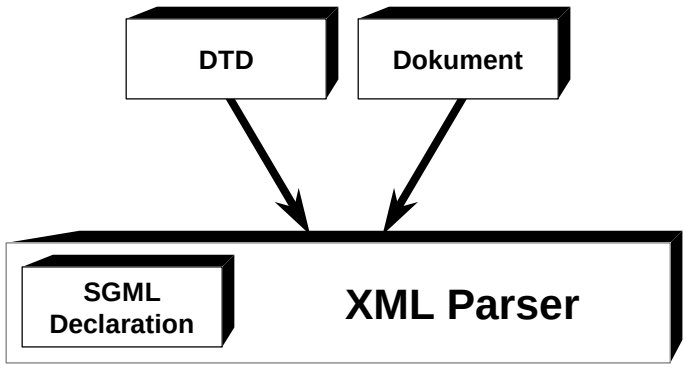
Entities

- das grundlegende Konstrukt zur Strukturierung
  - jedes “Stück XML Text” ist ein Entity
  - Entities werden deklariert (wie Elemente/Attribute)
- Benutzung in XML über spezielle Konstrukte
  - benutzt durch eine *entity reference* (`&entity;`)
  - XML Prozessor fügt den *replacement text* ein
- verschiedene Entity-Typen
  - *Parameter* (nur in DTDs verwendet)
  - *Internal General* (“Text-Makros” in Dokumenten)
  - *External Parsed General* (externe XML Ressourcen)
  - *Unparsed* (externe nicht-XML Ressourcen)
- vordefinierte Entities für Markup-Zeichen

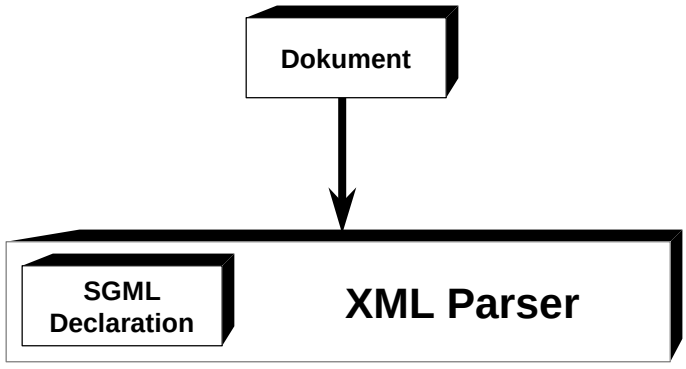
Well-formed und *valid* XML

- XML unterscheidet zwischen zwei “Levels”
  - *well-formed* gehorchen dem XML-Standard
  - *valid* sind well-formed und gehorchen einer DTD
- *well-formed* Dokumente
  - falls keine DTD vorhanden (nicht immer nötig!)
  - falls DTD nicht verfügbar
  - falls keine Weiterverarbeitung notwendig
- *valid* Dokumente
  - Validierung anhand einer DTD
  - notwendig zur Weiterverarbeitung
  - im B2B Umfeld wohl ausnahmslos valid XML

XML Parser (Validierung)



XML Parser (Well-formedness)



"Character Sets"

- der Begriff sollte vermieden werden!
- besser definierte Begriffe sind
  - *Character Repertoire* (Menge an Zeichen)
  - *Character Code* (Definition von *Code Points*)
  - *Character Encoding* (Codierung für Code Points)
- 7-bit ASCII (ISO 646) definiert alles zusammen
- ISO 8859 ist ASCII auf 8-bit erweitert
  - Varianten für verschiedene Erweiterungen
- Unicode trennt die verschiedenen Schritte
  - ISO 10646 und Unicode sind harmonisiert
  - Unicode definiert verschiedene Encodings

XML und Zeichensätze

- allgemeines Problem in der Informatik
  - im W3C bearbeitet im Bereich "I18N"
  - verbreitete Standards sind ASCII und ISO 8859-1
  - Benachteiligung vieler anderer Sprachen
- XML ist ein zeichenbasiertes Format
  - Problem bekannt von HTML (Umlaute)
  - XML verwendet Unicode als Default
- Unicode ist legt eine Menge von Zeichen fest
  - Zeichen identifiziert über *Code Points*
  - ein spezifisches Encoding legt die binäre Form fest
  - meistens UTF-8 (jedes ASCII-Dokument ist UTF-8!)

XML: Daten oder Dokumente?

- Dokumente sind zwar Daten, aber Daten nicht unbedingt Dokumente...
- XML stammt von SGML ab
  - SGML kommt aus dem Bereich der Dokumente
  - XML wird zunehmend für Daten eingesetzt
- XML ist stark in einigen Bereichen
  - semi-strukturierte Daten (Dokumente...)
- XML ist schwach in anderen Bereichen
  - Typ- und Vererbungskonzepte für Elemente
  - Datentypkonzepte überhaupt (kaum Datentypen)

Schema-Sprachen für XML

- DTDs sind der traditionelle Weg
  - definiert im SGML- und im XML-Standard
  - momentan der einzige etablierte Standard für Schemas
- DTDs haben einige Nachteile
  - die Syntax ist nicht XML-Dokumentensyntax
  - die Strukturierungsmechanismen sind einfach (insbesondere ist keine Vererbung möglich)
  - sie kennen keine Datentypen (schlecht für B2B Szenarien)
- weitergehende Ansätze
  - verwenden meistens XML-Syntax
  - bieten weitergehende Modellierungsmethoden
  - unterstützen Datentypen

Die Schema Begriffsverwirrung...

- eine Schema beschreibt eine Dokumentenklasse
  - Beschreibung von Element- und Attributtypen
  - Beschreibung ihrer erlaubten Verwendung
    - Kombinationsmöglichkeiten
    - erlaubte Datentypen in den Instanzen
- DTDs sind eine mögliche Schema-Sprache
  - speziell, weil im XML-Standard selbst definiert
- XML Schema vom W3C definiert
  - extrem unglückliche Namensgebung
- keine relevanten anderen Dialekte

W3C XML Schema

- zweigeteilter Standard
  - ein Standard für die Strukturierung
  - ein Standard für die Datentypen
- DTDs decken nur den ersten Teil ab
- Implementierungen vorhanden
  - Microsofts *XML-Data* ist ein proprietärer Ansatz
  - *Document Content Description (DCD)*
- momentaner Status *Working Draft* (02/00)

XML Schema Part 1: Structures

- Reformulierung der DTD-Mechanismen in XML
  - Elemente zur Elementbeschreibung
  - Elemente zur Elementdefinition (+ Content Model)
  - Elemente zur Attributdefinition
- Tools zur Konvertierung von DTDs nach Schema
  - zu wenig Infos (Schema ist mächtiger als DTD)
  - Rückrichtung möglich, aber nur mit Verlusten
- Modellierung immer mit Blick auf Schema
  - DTD als Zwischenlösung, Schema als Grundlage
- Tools unterstützen manchmal nur DTDs
  - Frage: Validierung gemäss DTD oder Schema?

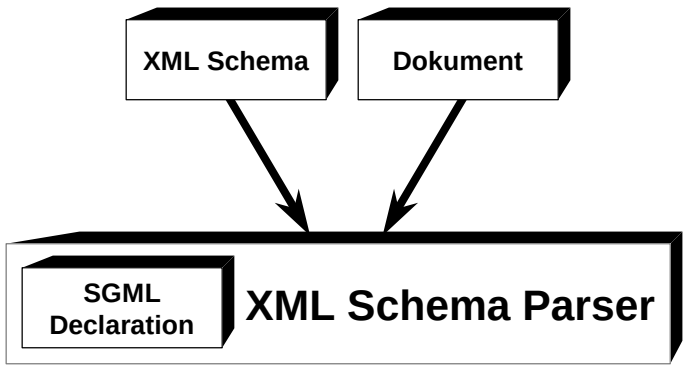
XML Schema Part 2: Datatypes

- definiert ein Typensystem für XML Schema
  - einige Grundtypen (Zahlen, Strings, Datum, …)
  - benutzerdefinierte Typen
- Datentypen werden charakterisiert
  - Einschränkungen auf Wertebereiche
  - Einschränkungen auf lexikalische Werte
  - Verwendung von *Regular Expressions*
- Implementierungen bisher keine!
  - grosser Aufwand (recht komplexer Standard)
  - grosser Nutzen (hohe Qualität der Dokumente)
- Standard verfolgen und als Ziel sehen

Valid und *schema-valid* XML

- XML unterscheidet zwischen zwei "Levels"
  - *well-formed* gehorchen dem XML-Standard
  - *valid* sind well-formed und gehorchen einer DTD
- *well-formed* und *valid* Konzepte
  - sind direkt im XML Standard definiert
  - können mit DTD und Dokument verifiziert werden
- *schema-valid* Dokumente
  - müssen gemäss eines XML Schema validiert werden
  - gibt es nur mit XML Schema Applikationen
  - haben mehr Randbedingungen als *valid* Dokumente
  - sollten kontrolliert importiert/exportiert werden

XML Schema Parser



Was XML alles nicht kann...

- XML ist ein textorientiertes Format
  - keine beliebigen Datentypen
  - keine effiziente Speicherung von Datentypen
  - Einbindung von BLOBs über Referenzen
- XML ist kein kompaktes Format
  - Speicherung alles andere als effizient
  - selbstbeschreibende Daten als Hauptziel
- XML bietet keine Datenaustauschformate
  - Einigung auf gemeinsame Formate notwendig
  - auch XML-Anwendungen können inkompatibel sein
- XML braucht eine passende Umgebung

Zusammenfassung

- XML als Syntax für strukturierte Daten
  - Grammatik wird in DTD beschrieben
  - hierarchische Organisation der Daten
  - Strukturierung in Elemente/Attribute
  - Unterscheidung *well-formed/valid* XML Dokumente
- XML Schema als neue "DTD"
  - wesentlich besser für B2B geeignet als DTDs
  - mittelfristig Ablösung der DTDs