Einführung in XML

Felix Sasaki

Version: November 2016

Vorspann

Ziel des Seminars

- In die Technologie XML einführen
  - Hintergründe
  - Anwendungen
  - Aspekte
  - Gegenwart und Ausblick
- Relevanz für Archivare aufzeigen
- Praxisbeispiele nachvollziehbar darstellen
- „Lust auf mehr“ fördern

Über den Dozenten

- „Senior Researcher“ am DFKI (Deutsches Forschungszentrum für künstliche Intelligenz)
- Mitarbeiter beim W3C (World Wide Web Consortium)
- Schwerpunkt: Mehrsprachigkeit und bzw. in Webtechnologien, insbesondere XML und Semantic Web

Über Sie!

- Wer sind Sie?
- Was wissen Sie schon über XML?
- Was haben Sie für Erwartungen an das Seminar?

Überblick

- Einführung
- XML – Geschichte und Anwendungsszenarien
- Das XML-Universum
- Anwendungen von XML für Archivare
- XML – Zukunft

XML steht für

- eXtensible Markup Language (erweiterbare Auszeichnungssprache)
- Ein Standard des World Wide Web Consortium (W3C), 1998 veröffentlicht
- Was bedeutet „erweiterbare Auszeichnungssprache“?

Was haben XML und Flugzeuge miteinander zu tun?

Flugzeuge brauchen gute Anleitungen (Teil 1)

- Flugzeuganleitungen sind komplex und umfangreich (> 100.000 Seiten Text)
- Konsistenz in Dokumentstruktur, Terminologie, Verweisen etc. ist überlebenswichtig

➔ der Bereich „technische Dokumentation“ braucht Hilfsmittel, um diese Konsistenz sicher zu stellen

Text + Auszeichnung

- Auszeichnung hilft Konsistenz zu wahren
- Beispiel: Fest definierte Terme sind unterstrichen

„Der Treibstofftank besteht aus diesen Konstruktionselementen: ...“

Text + Auszeichnung

- „Unterstrichen“ ist eine Darstellung für den menschlichen Leser
- Die Basis ist für den Computer lesbare Auszeichnung (hier via „<term>...</term>“)

„Der Treibstofftank besteht aus diesen Konstruktionselementen: ...“
„Der <term>Treibstofftank</term> besteht aus diesen <term>Konstruktionselementen</term>: ...“

Flugzeuge brauchen gute Anleitungen (Teil 2)

- Hersteller von technischen Dokumentationen waren eine treibende Kraft bei der Entwicklung von Auszeichnungssprachen
- Eine weitere Auszeichnungssprache: LaTex
  - Auszeichnungssprache / Textsatzsystem
  - Hauptzweck: wissenschaftliche Publikationen

LaTex - Beispiel

- Auszeichnung von
  - Metadaten für das gesamte Dokument
  - Dokumentstruktur
  - Inhalten

```
\documentclass[12pt]{article}
\title{\LaTeX}
\begin{document}
  \maketitle
  \section{Introduction}
  This document describes the \textbf{document preparation system} called
     \LaTeX{}. ...
\end{document}
```

Arten von Auszeichnungen

- Präsentationsbezogen („dieses Wort ist im Fettdruck“)
- Inhaltsbezogen („dieses Wort ist ein vordefinierter Term“)
- Manche Auszeichnungssprachen mischen Auszeichnung von Präsentation und Inhalt (z.B. LaTex)

Vorteile der Trennung von inhalts- und darstellungsbezogener Auszeichnung

- Arten inhaltsbezogener Auszeichnung sind eher konstant als Darstellung
  - Darstellung eines Terms „unterstrichen“ im Web
  - Darstellung eines Terms „im Fettdruck“ bei gedruckter Ausgabe
- Wiederverwendung von Text + Auszeichnung ist einfacher wenn diese „nur“ inhaltsbezogen ist
- Single-Source-Publishing (eine Quelle, mehrere Ausgabeformate) ist leichter realisierbar

Flugzeuge brauchen gute Anleitungen (Teil 3)

- (Nicht nur) Flugzeughersteller brauch(t)en eine Auszeichnungssprachen zur inhaltlichen (nicht darstellungsbezogenen) Auszeichnung
- XML ist das Ergebnis entsprechender Standardisierungsbestrebungen

Vorläufer von XML

- GML
  - Generalized Markup Language, von IBM entwickelt
- SGML
  - Standard GML, ISO 8879:1986 SGML
  - Sehr komplex, nie komplett umgesetzt
- XML 1.0
  - Eine Teilmenge von XML (SGML minus vieler komplexer Bestandteile)
- XML basierte Technologien (vgl. „Das XML-Universum)

Unser erstes XML-Dokument

```
<flugzeugAnleitung version="1.0">
 <metaInformation>...</metaInformation>
 <text>
  <abschnitt>
   <paragraph>...</paragraph>
  </abschnitt> ...
 </text>
</flugzeugAnleitung>
```

Vgl. beispiele/flugzeuganleitung.xml

Demo / Übung: XML-Dokument im Browser

Unser erstes XML-Dokument

- Auszeichnung durch Elemente („flugzeugAnleitung“, „metaInformation“, „text“, ...)
- Elemente haben Starttag (<text>) und Endtag (</text>)

```
<flugzeugAnleitung version="1.0">
 <metaInformation>...</metaInformation>
 <text>
  <abschnitt>
   <paragraph>...</paragraph>
  </abschnitt> ...
 </text>
</flugzeugAnleitung>
```

Unser erstes XML-Dokument

- Dokumentstruktur: ein Baum
  - Ein Wurzelelement (hier „flugzeugAnleitung“)
  - Hierarchisch untergeordnete Elemente und Textinhalte
  - Nicht-hierarchische Strukturierung (z.b. „<abschnitt> <paragraph> </abschnitt> <paragraph>“) ist verboten

```
<flugzeugAnleitung version="1.0">
    <metaInformation>...</metaInformation>
        <text>
            <abschnitt>
                <paragraph>...</paragraph>
            </abschnitt> ...
        </text>
</flugzeugAnleitung>
```

Unser erstes XML-Dokument

- Zusatzinformationen durch Attribute
  - Werden im Starttag eines Elements geschrieben
  - Null bis beliebig viele Attribute
  - Haben den Aufbau „Name – Gleichheitszeichen – Wert in Anführungszeichen“
  - Ein Attributname kann am gleichen Element nur einmal vorkommen

```
<flugzeugAnleitung version="1.0">
 <metaInformation>...</metaInformation>
 <text>
  <abschnitt>
   <paragraph>...</paragraph>
  </abschnitt> ...
 </text>
</flugzeugAnleitung>
```

Unser erstes XML-Dokument

- Wer entscheidet über Namen?
  - Man selbst: In XML kann man Namen von Elementen und Attributen selbst vergeben – Deshalb ist XML erweiterbar (eXtensible), anders als z.B. LaTex
  - Es gibt Möglichkeiten durch validieren die „richtigen Namen“ für eine Anwendung sicher zu stellen (dazu später mehr)

```
<flugzeugAnleitung version="1.0">
 <metaInformation>...</metaInformation>
 <text>
  <abschnitt>
   <paragraph>...</paragraph>
  </abschnitt> ...
 </text>
</flugzeugAnleitung>
```

Fehler von „Wohlgeformtheit“ in XML-Dokumenten

- Beschriebene Regeln für „Wohlgeformtheit“ (den Dokumentaufbau) müssen eingehalten werden
  - Dokumentaufbau: ein Wurzelelement, hierarchischer Aufbau
  - Spitze Klammern nicht vergessen
  - Anführungsstriche bei Attributen nicht vergessen
  - …
- Fehler werden drakonisch geahndet

Demo / Übung: Fehler im XML-Dokument einbauen

Weitere Infos

- Vgl.

http://www.teialehrbuch.de/Kostenlose-Kurse/XML/7691-Der-Aufbau-eines-XML-Dokuments.html

Anwendungsszenarien für XML

- Technische Dokumentation (vgl. Flugzeugbeispiel, Staubsaugeranleitung, ...)
- Andere Arten von Publikationen (z.B. wissenschaftlicher Bereich) / Textaufbereitung

Demgegenüber

- XML im Bankautomaten, an der Tankzapfsäule, für bibliographische oder andere, verschiedenste Arten von Datensätzen

Was ist der Unterschied?

Dokumentorientieres versus datenorientiertes XML

- Zweck: Auszeichnen von Texten
- Aufbau: meist Dokumentstruktur (vgl. „Unser erstes XML-Dokument“)
- Nutzung:
  - Wiederverwendbarkeit (z.B. für Single-Source Publishing) von Texten

- Zweck: Auszeichnen von Daten
- Aufbau: Eine Folge von Datensätzen:
  - Datensatzsammlung
    - Datensatz
      - Feld, Wert
      - …
- Nutzung:
  - XML als allgemeines Datenaustauschformat

Beispiel für datenorientiertes XML

- Katalog für Flugzeugerstatzteile
- Wurzelelement „Katalog“
- Für jeden Eintrag ein „eintrag“ Element
- Für jedes Feld ein untergeordnetes Element (z.B. „name“) mit jeweiligem Wert

```
<katalog typ="Ersatzteilliste">
 <eintrag nummer="1">
  <name>Tragfläche linke Seite</name>
  <status>verfügbar</status>
 </eintrag> ...
</katalog>
```

Vgl. beispiele/katalog.xml

Warum datenorientiertes XML?

- Vor XML
  - Speicherung, Verarbeitung und Weitergabe strukturierter Daten waren spezifisch für Anwendungen und Programmiersprachen
    - SQL-Datenbank, Java Resource Bundle, Excel-Tabellenblatt, MARC21-Bibliothekskatalog, Infodata-Thesaurus, ...
- Vorteil von XML
  - Daten und Anwendung sind entkoppelt und leicht wieder verwendbar / neu kombinierbar
    - Ersatzteilkatalog zur Suche, Ersatzteilkatalog als Teil einer Dokumentation, Ersatzteilkatalog von Firma A in System der Firma B integriert, ...

Was für wen?

- Für Archivare ist datenorientieres XML am relevantesten
  - Beispiel: Findbücher sind eher Datensätze als Textdokumente
- XML-Anwendungen im Archivbereich (vgl. späterer Abschnitt) demonstrieren dies

Flugzeuge brauchen gute Anleitungen (Teil 4 und Schluss)

- Die Nutzung von XML als Datenaustauschformat war nicht „vorgesehen“
- Hauptmotivation war die beschriebene dokumentorientierte Anwendung
- In der Gegenwart gibt es beide Bereiche in großem Maße

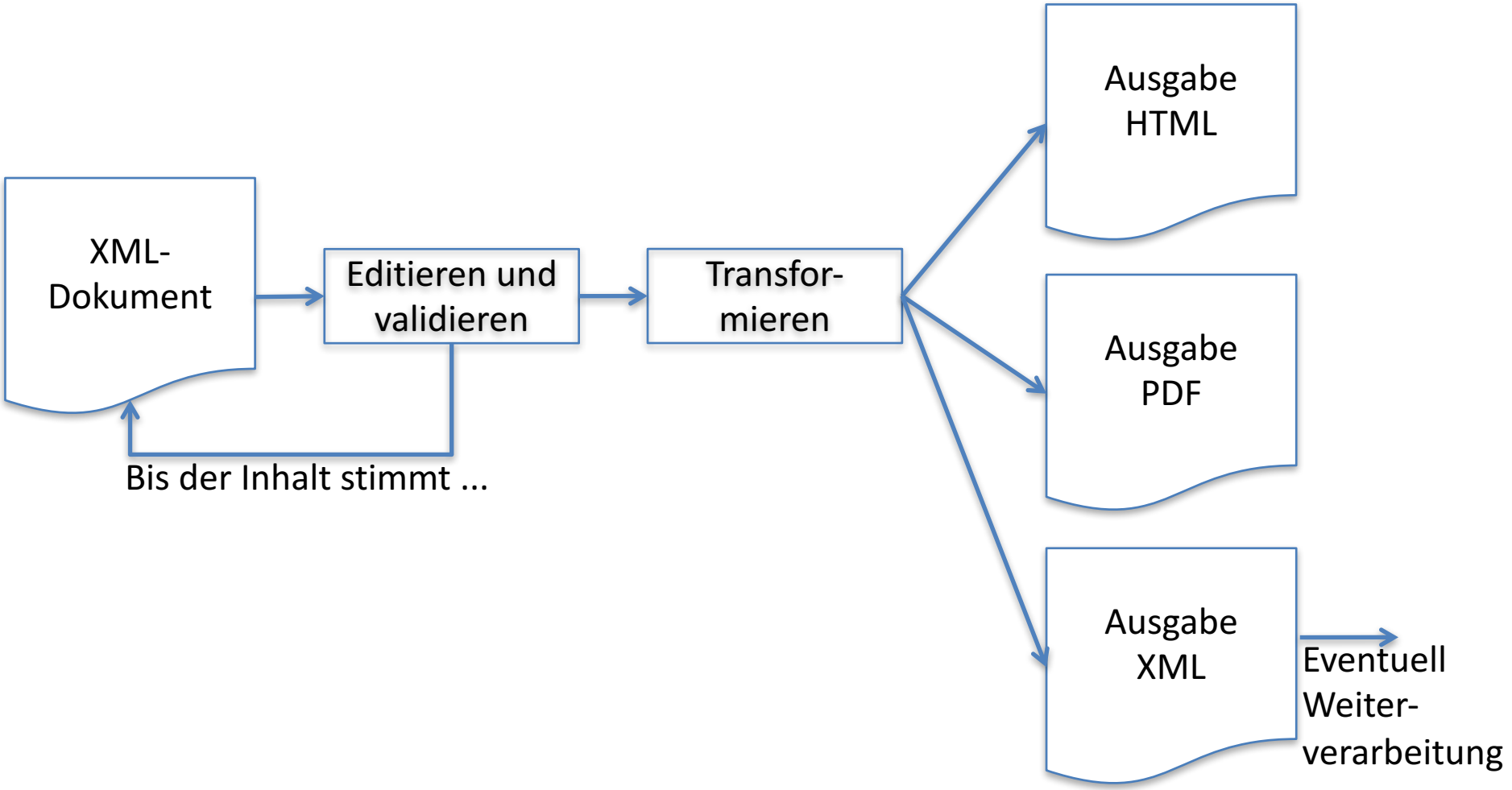
Überblick

- Einführung
- XML – Geschichte und Anwendungsszenarien
- Das XML-Universum
- Anwendungen von XML für Archivare
- XML – Zukunft

Das XML-Universum: XML-Technologien

- XML 1.0 und XML Namensräume
- XML Validierung: Schemasprachen
  - XML Schema, RELAX NG, Schematron, ...
- XML Transformation und Abfrage
  - XPath, XSLT, XQuery, ...
- Verschiedene XML-Anwendungen

Das große Bild: Prototypische Verlauf einer XML-Verarbeitung



Was muss ich können?

- Als XML-Anwender: oft nichts
  - Viele XML-Formate sind „verborgen“ (Beispiel: diese Präsentation ist ein XML-Format!)
- Als XML „Autor“
  - XML 1.0 und Namensräume zum Editieren von Dokumenten
- Als Entwickler von XML-Anwendungen
  - Auch den Rest ☺ - was genau, hängt vom Anwendungszweck ab

XML 1.0 und Namensräume

- XML 1.0: Definiert beschriebenen Aufbau von XML-Dokumenten (Wiederholung)

```
<flugzeugAnleitung version="1.0">
 <metaInformation>...</metaInformation>
 <text>
  <abschnitt>
   <paragraph>...</paragraph>
  </abschnitt> ...
 </text>
</flugzeugAnleitung>
```

XML 1.0 und Namensräume

- Namensräume: können dazu dienen, XML-Elemente und Attribute zu differenzieren, d.h. in „XML Vokabulare“ zu unterscheiden
- Anwendung ist nicht zwingend – vgl. bisherige Beispiele
- Beispiel für Kombination von XML Vokabularen in einem Dokument: Katalog enthält eine Visualisierung unter Nutzung des SVG Vokabulars, d.h. mit Elementen aus dessen Namensraum
- Ein Namensraum wird durch das Attribut „xmlns“ definiert. Es ordnet das Namensraumpräfix (vom Nutzer frei wählbar) der Namensraum-URI (für das jeweilige Vokabular fest definiert) zu.

```
<katalog typ="Ersatzteilliste">
 <eintrag nummer="1">
  <name>Tragfläche linke Seite</name>
  <g:svg xmlns:g="http://www.w3.org/2000/svg">...</g:svg>
  <status>verfügbar</status>
 </eintrag> ...
</katalog>
```

Vgl. beispiele/katalog-mit-namensraeumen.xml

Namensräume – Anlass zu Fehlern ...

- Den Überblick behalten
  - Wo ist der Namensraum definiert?
  - Wo gilt welches Präfix?
  - Wo gilt das Präfix nicht?

Beispiel und Demo: Fehler mit Namensräumen

```
<katalog typ="Ersatzteilliste“> ... <name>Tragfläche linke Seite</name>
   <svg:svg xmlns:svg="http://example.com/my-namespace">tbd</svg:svg>
   <status>verfügbar</status>
   <zweitansicht>
     <svg:svg>tbd</svg:svg>
   </zweitansicht>
   </eintrag> ...
</katalog>
```

Vgl. beispiele/katalog-mit-namensraeumen-fehlerhaft.xml

Weitere Infos

- Vgl.

http://www.teialehrbuch.de/Kostenlose-Kurse/XML/7699-Namensraeume.html

Exkurs: XML-Tools

- Kommerzielle Editoren
  - XMLSpy besonders für datenorientiertes XML
  - oXygen für dokumentorientiertes XML
- Freie Tools
  - EditiX erlaubt grundlegende XML Verarbeitung (validieren, transformieren)
  - Grundlage der folgenden Demos

XML Validierung

- Zweck: sicherstellen dass ein Dokument die für das Vokabular „richtigen“ Elemente und Attribute enthält
  - Ein „katalog“ Element (kein „katalg“)
  - Mindestens ein „eintrag“ Element mit obligatorischem „nummer“ Attribut
  - Ein „name“ Element, gefolgt von optional einem „svg“ Element, gefolgt von einem „status“ Element, ...

```
<katalog typ="Ersatzteilliste">
 <eintrag nummer="1">
  <name>Tragfläche linke Seite</name>
  <g:svg xmlns:g="http://www.w3.org/2000/svg">...</g:svg>
  <status>verfügbar</status>
 </eintrag> ...
</katalog>
```

XML Validierung: Methoden

- Validierung via so genannter Schemasprachen
- Die wichtigsten:
  - XML DTD, vgl. beispiele/katalog.dtd
  - XML Schema, vgl. beispiele/katalog.xsd
  - RELAX NG, vgl. beispiele/katalog.rng bzw. katalog.rnc
  - Schematron, hier ohne Beispiel
- Die Unterschiede sind hier irrelevant
- Devise: Keine eigenen XML-Vokabulare definieren, sondern bestehende. Dann validiert man mit dem Schema, welches zur Verfügung steht

Demo: Validierung einer XML-Datei

- Funktioniert nicht in herkömmlichen Browsern
- Diese prüfen nur die Wohlgeformtheit von XML-Dokumenten

Vgl. beispiele/katalog-mit-interner-dtd.xml

XML-Datei mit interner DTD (entspricht katalog.dtd)

```
<!DOCTYPE katalog [
<!ELEMENT katalog (eintrag+)>
<!ATTLIST katalog
  typ NMTOKEN #REQUIRED>

<!ELEMENT eintrag (name,status)>
<!ATTLIST eintrag
  nummer CDATA #REQUIRED>

<!ELEMENT name (#PCDATA)>
<!ATTLIST name>

<!ELEMENT status (#PCDATA)>
<!ATTLIST status>
]>
<katalog typ="Ersatzteilliste">
 <eintrag nummer="1">
   <name>Tragfläche linke Seite</name>
   <status>verfügbar</status>
 </eintrag>
</katalog>
```

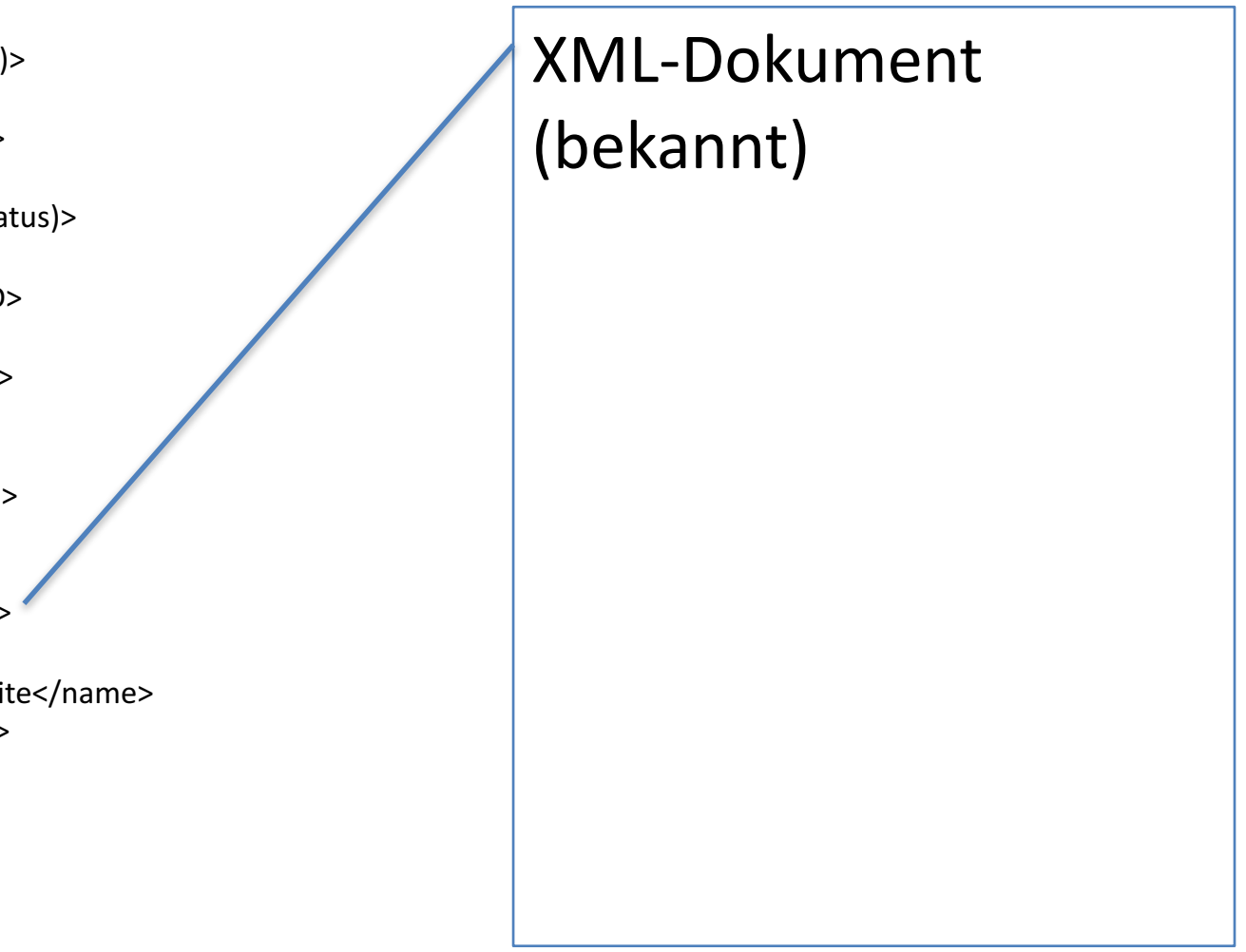
XML-Datei mit interner DTD

```
<!DOCTYPE katalog [
<!ELEMENT katalog (eintrag+)>
<!ATTLIST katalog
  typ NMTOKEN #REQUIRED>

<!ELEMENT eintrag (name,status)>
<!ATTLIST eintrag
  nummer CDATA #REQUIRED>

<!ELEMENT name (#PCDATA)>
<!ATTLIST name>

<!ELEMENT status (#PCDATA)>
<!ATTLIST status>
]>
<katalog typ="Ersatzteilliste">
 <eintrag nummer="1">
   <name>Tragfläche linke Seite</name>
   <status>verfügbar</status>
 </eintrag>
</katalog>
```



XML-Datei mit interner DTD

```
<!DOCTYPE katalog [
<!ELEMENT katalog (eintrag+)>
<!ATTLIST katalog
  typ NMTOKEN #REQUIRED>

<!ELEMENT eintrag (name,status)>
<!ATTLIST eintrag
  nummer CDATA #REQUIRED>

<!ELEMENT name (#PCDATA)>
<!ATTLIST name>

<!ELEMENT status (#PCDATA)>
<!ATTLIST status>
]>
<katalog typ="Ersatzteilliste">
 <eintrag nummer="1">
  <name>Tragfläche linke Seite</name>
  <status>verfügbar</status>
 </eintrag>
</katalog>
```

Einbettung der DTD mit Festlegung des Wurzelelements

XML-Datei mit interner DTD

```
<!DOCTYPE katalog [
<!ELEMENT katalog (eintrag+)>
<!ATTLIST katalog
  typ NMTOKEN #REQUIRED>

<!ELEMENT eintrag (name,status)>
<!ATTLIST eintrag
  nummer CDATA #REQUIRED>

<!ELEMENT name (#PCDATA)>
<!ATTLIST name>

<!ELEMENT status (#PCDATA)>
<!ATTLIST status>
]>
<katalog typ="Ersatzteilliste">
 <eintrag nummer="1">
  <name>Tragfläche linke Seite</name>
  <status>verfügbar</status>
 </eintrag>
</katalog>
```

Elementdeklaration (exemplarisch)
Bedeutet: „Element ‚katalog‘ besteht aus mindestens einem ‚eintrag‘ Element“

XML-Datei mit interner DTD

```
<!DOCTYPE katalog [
<!ELEMENT katalog (eintrag+)>
<!ATTLIST katalog
  typ NMTOKEN #REQUIRED>

<!ELEMENT eintrag (name,status)>
<!ATTLIST eintrag
  nummer CDATA #REQUIRED>

<!ELEMENT name (#PCDATA)>
<!ATTLIST name>

<!ELEMENT status (#PCDATA)>
<!ATTLIST status>
]>
<katalog typ="Ersatzteilliste">
 <eintrag nummer="1">
  <name>Tragfläche linke Seite</name>
  <status>verfügbar</status>
 </eintrag>
</katalog>
```

Attributdeklaration(exemplarisch)
Bedeutet: „Element ‚katalog‘ hat obligatorisches Attribut ‚typ‘; Attributwert darf nur bestimmte Zeichen enthalten (z.B. keine Leerzeichen)“

Weitere Infos

- Vgl.

http://www.teialehrbuch.de/Kostenlose-Kurse/XML/7730-Die-Dokumenttyp-Definition.html

XML Transformation und Abfrage

- XSLT: Transformation von XML-Dokumenten
  - Nach HTML
  - Nach PDF
  - In andere XML-Dokumente (z.B. um verschieden strukturierte Daten zusammen zu führen)
  - …
- XQuery
  - Abfragesprache für XML
  - Vergleichbar zu SQL für relationale Datenbanken

XML Transformation und Abfrage

- Grundlage beider: XPath
  - Mittel um Teile eines XML-Dokumentes zu selektieren
    - Alle Datensätze
    - Elemente im ersten Datensatz
    - Datensätze vom Typ „Turbine“
    - ...
  - ... zur weiteren Verarbeitung (Transformation, Abfrage)

Demo: XPath

- Demo im EditiX Editor:
  - Selektion des „katalog“ Elements: /katalog
  - Selektion der „eintrag“ Elemente: /katalog/eintrag
  - Selektion der „eintrag“ Elemente mit dem Status „verfügbar“: katalog/eintrag[status='verfügbar']

Typische XPath-Ausdrücke: Vgl.

http://www.w3.org/TR/xpath/#path-abbrev

Vgl. beispiele/katalog.xml

Transformationsbeispiel / Demo

- Eingabe: katalog.xml
- Ausgabe: katalog.html (HTML Darstellung des Katalogs)
- XSLT-Datei: katalog-nach-html-1.xsl
- Vorgehen:
  - Saxon Prozessor saxon9he.jar in Verzeichnis „beispiele“ kopieren
  - Eingabeaufforderung öffnen
  - Eingabe (in einer Zeile, dann RETURN):

```
java -cp saxon9he.jar net.sf.saxon.Transform katalog.xml Katalog-nach-html-1.xsl > katalog.html
```

Vgl. beispiele/katalog-nach-html-1.xsl

Transformationsbeispiel / Demo

- Eingabe: katalog.xml
- Ausgabe: katalog-analyse.html (Analyse und HTML Darstellung des Katalogs)
- XSLT-Datei: katalog-nach-html-2.xsl
- Eingabe (in einer Zeile, dann RETURN):

java -cp saxon9he.jar net.sf.saxon.Transform katalog.xml Katalog-nach-html-2.xsl > katalog-analyse.html

Vgl. beispiele/katalog-nach-html-2.xsl

Weitere Infos

- Vgl.

http://www.teialehrbuch.de/Kostenlose-Kurse/XML/7785-XSLT-und-XPath.html

XQuery-Abfrage / Demo

- Eingabe: „nur“ XQuery-Datei katalog-abfrage.xql
- Daten sind Teil der XQuery-Datei (können auch separat stehen)
- Ausgabe: katalog-abfrage-ergebnis.txt
- Vorgehen
  - Saxon Prozessor saxon9he.jar in Verzeichnis „beispiele“ kopieren
  - Eingabeaufforderung öffnen
  - Eingabe (in einer Zeile, dann RETURN):

```
java -cp net.sf.saxon.Query katalog-abfrage.xql > katalog-abfrage-ergebnis.txt
```

Vgl. beispiele/katalog-abfrage.xql

Beispiele für XML-Anwendungen

TEI

- „Text Encoding Initiative“
- XML-Vokabular (und Organisation) zur Kodierung von geisteswissenschaftlichen Texten
  - Prosa
  - Drama
  - Transkription gesprochener Sprache
  - ...
  - digitalisierte Texte
- Häufige Anwendung in digitalen Archiven

Struktur eines TEI-Dokuments (TEI Version P5)

- Wurzelelement „TEI“
- „teiHeader“: Metainformationen
  - „fileDesc“: Zur TEI-Datei selbst
  - „encodingDesc“: Verhältnis zur Quelle (z.B. der Digitalisierung)
  - „profileDesc“: nicht-bibliographische Aspekte (Sprache(n), beteiligte Personen, ...)
  - „revisionDesc“: Revision
- „text“: eigentlicher Inhalt

Vgl. beispiele/tei-beispiel.xml

Demo: Transformation nach HTML

- Stylesheet: standardisiertes Stylesheet, vgl.

http://www.tei-c.org/release/xml/tei/stylesheet/xhtml2/tei.xsl

- Eingabe: tei-beispiel.xml
- Ausgabe: tei-beispiel.html
- Eingabe (in einer Zeile, dann RETURN):

```
java -cp saxon9he.jar net.sf.saxon.Transform tei-beispiel.xml
http://www.tei-c.org/release/xml/tei/stylesheet/xhtml2/tei.xsl
> tei-beispiel.html
```

TEI – ein Pool von Modulen

- ... für Prosa, Transkription, ...
- Nutzer stellen sich die notwendigen Module (Schemas + Dokumentation) selbst zusammen
- Vgl. http://www.tei-c.org/Roma/

Beispielprojekte für TEI

- Deutsches Textarchiv
  - Vgl. http://www.deutschestextarchiv.de/
  - Deutschsprachige Texte ca. 1650 – 1900
  - Hauptsächlich Nutzung durch Linguisten
- CESG
  - Vgl. http://www.cesg.unifr.ch/de/beschreibung.htm
  - Erschließung frühneuzeitlicher Handschriften mit TEI

SVG

- Standardized Vector Graphics
- Zweck: Repräsentation (interaktiver) Vektorgraphiken
- Anwendung: oft als Bestandteil anderer Technologien
  - Beispiel: Anwendung in TEI

Demo: SVG in TEI

- TEI Dokument mit SVG wird nach HTML transformiert
- Stylesheet: standardisiertes Stylesheet, vgl.

http://www.tei-c.org/release/xml/tei/stylesheet/xhtml2/tei.xsl

- Eingabe: tei-plus-svg.xml
- Ausgabe: tei-plus-svg.html

Eingabe (in einer Zeile, dann RETURN):

```
java -cp saxon9he.jar net.sf.saxon.Transform tei-plus-svg.xml
http://www.tei-c.org/release/xml/tei/stylesheet/xhtml2/tei.xsl
> tei-plus-svg.html
```

- Darstellung einer Transformation in manchen Browsern möglich

Vgl. beispiele/tei-plus-svg.xml

MathML

- Zweck: Repräsentation mathematischer Formeln via XML
- Anwendung: oft als Bestandteil anderer Technologien
  - Auch hier: Anwendung in TEI

Demo: Transformation nach HTML

- TEI Dokument mit MathML wird nach HTML transformiert
- Stylesheet: standardisiertes Stylesheet, vgl.

http://www.tei-c.org/release/xml/tei/stylesheet/xhtml2/tei.xsl

- Eingabe: tei-plus-mathml.xml
- Ausgabe: tei-plus-mathml.html

Eingabe (in einer Zeile, dann RETURN):

java -cp saxon9he.jar net.sf.saxon.Transform tei-plus-mathml.xml http://www.tei-c.org/release/xml/tei/stylesheet/xhtml2/tei.xsl > tei-plus-mathml.html

- Darstellung einer Transformation in manchen Browsern möglich

Vgl. beispiele/tei-plus-mathml.xml

DocBook

- XML-Vokabular für technische Dokumentation
- Prototypische Lösung für unser Problem „konsistente Flugzeuganleitungen“
- Ein weiteres Vokabular: DITA (hier nicht behandelt)
- Die Qual der Wahl: welches von beiden verwenden?
  - Keine einfache Antwort
  - Abhängig von Rahmenbedingungen wie Projektzusammenhang, verfügbare Tools, eigene Kenntnisse, ...

DocBook Beispiel

- Hier: wenig Metadaten („info“ Element)
- Viele aus HTML bekannte Elemente
- Mehr Strukturierung als in HTML möglich bzw. erforderlich

Vgl. beispiele/docbook-beispiel.xml

Demo: Transformation nach HTML

- Standardisierte Stylesheets: vgl.

http://norman.walsh.name/2011/08/25/docbook-xslt-2

- Auch Teil der Kursunterlagen
- Eingabe: docbook-beispiel.xml
- Ausgabe: docbook-beispiel.html

Eingabe (in einer Zeile, dann RETURN):

```
java -cp saxon9he.jar net.sf.saxon.Transform docbook-beispiel.xml
docbook-xslt2-2.0.0/xslt/base/html/docbook.xsl > docbook-
beispiel.html
```

- XSLT-Datei:

docbook-xslt2-2.0.0/xslt/base/html/docbook.xsl

Überblick

- Einführung
- XML – Geschichte und Anwendungsszenarien
- Das XML-Universum
- Anwendungen von XML für Archivare
- XML – Zukunft

EAD

- „Encoded Archival Description“
- XML-basierter Standard der Library of Congress für Findbücher
- Prototypischer Aufbau vgl. ead.xml

Struktur eines EAD-Dokuments

- Wurzelelement „ead“
- „eadheader“: Metainformationen
  - „eadid“: Identifikator des Findbuchs
  - „filedesc“: Zur EAD-Datei selbst
  - „profiledesc“: zum Findbuch (Erstellungsdatum, Sprache, ...)
  - „revisiondesc“: Revision
- „archdesc“: Beschreibung

Vgl. beispiele/ead.xml

Inhalte der „archdesc“ (Ausschnitt)

- „did“: Beschreibung der Kollektion (Herkunft, Datum, physikalische Eigenschaften, ...)
- „phystech“: physikalischer Zustand
- „arrangement“: Anordnung der Materialien
- „prefercite“: Bevorzugte Art des Zitierens
- „custodhist“: Kette der Besitzerschaft / Provenanz
- „appraisal“: Archivischer Wert

Demo: Transformation nach HTML

- Wieder „standardisiertes“ Stylesheet:

http://nwda-db.wsulibs.wsu.edu/xsl/project.xsl

- Eingabe: ead.xml
- Ausgabe: ead.html

Eingabe (in einer Zeile, dann RETURN):

java -cp saxon9he.jar net.sf.saxon.Transform ead.xml
http://nwda-db.wsulibs.wsu.edu/xsl/project.xsl > ead.html

Achtung: Ausgabe ist projektspezifisch

Anwendung von EAD: daofind

- Vgl. http://www.bundesarchiv.de/daofind/
- Projekt des Bundesarchivs
- Softwarepacket
  - Erstellung von Online-Findbüchern
  - Nutzung weiterer Standards (METS, EAC)

Nicht für Archivare, aber relevant

- MARCXML, vgl. http://www.loc.gov/standards/marcxml//
- XML Format für bibliographische Datensätze
  - sandburg.mrc (Original MARC Record)
  - sandburg.xml (MARCXML Version)
  - sandburg.html (HTML Version)
  - sandburgmods.xml (MODS Version)
  - sandburgdc.xml (Dublin Core Version)

Vgl. beispiele/marcxml/

Nicht für Archivare, aber relevant

- MARCXML vs. MODS vs. Dublin Core
  - MARCXML: MARC Felder 1:1 vom MARC Format übernommen
  - MODS (Metadata Object Description Schema): XML Elemente haben explizite, detaillierte Semantic; gut lesbar für den Fachmann
  - Dublin Core: XML Elemente haben sehr allgemeine Semantik; gut lesbar für jedermann (aber nicht detailliert)

Vgl. beispiele/marcxml/

Nicht für Archivare, aber relevant

- MADS
  - Metadata Authority Description Schema
  - Definition kontrollierter Vokabulare für Personen, Institutionen, Publikationstitel, ...
  - Ebenfalls als XML Vokabular vorhanden
  - Weiterentwicklung zu RDF als „Linked Data“ Vokabular – nicht Thema in diesem Seminar ☺

Vgl. beispiele/mads/

Überblick

- Einführung
- XML – Geschichte und Anwendungsszenarien
- Das XML-Universum
- Anwendungen von XML für Archivare
- XML – Zukunft

Braucht man in 10 Jahren noch XML?

- XML wird in verschiedensten Bereichen genutzt
  - Vergleiche diese Präsentation für Beispiele
- Es wird aus der Infrastruktur zur Repräsentation und Verarbeitung strukturierter Informationen nicht so schnell verschwinden

XML ist „reif“

- XML-Technologien funktionieren mehr und mehr problemlos
- Notwendigkeit für Wartung / Veränderungen / völlig neue Systeme sinkt
- Notwendigkeit XML im Detail zu lernen sinkt
- XML wird allgegenwärtig ... und geht ins Verborgene
  - Vergleich: Wer kennt TCP/IP?

Beispiel „XML im Verborgenen“: XML in dieser Präsentation

- Benennen Sie die Datei xml-seminar.pptx in xml-seminar.zip um
- Entpacken Sie die Datei
- Der Inhalt: hauptsächlich eine Reihe von XML-Dateien!

Relevanz für Archivare?

- Mehr und mehr verbergen von XML-Strukturen in „schönen“ Benutzerinterfaces
- Details von XML sind für wenige Experten wichtig

Exkurs: XML im Web

Der Plan

HTML 4.01 (1999)

XHTML 1.0 (2000)

XML im Browser

XHTML 2.0

XHTML 2.0
CURIE
XFrames
HLink
XHTML+MathML+SVG Profile
XHTML Modularization 1.0 Second Edition

Die Realität

HTML 4.01 (1999)

XHTML 1.0 (2000)

HTML5

Warum gestoppt?

XHTML 2.0

XHTML 2.0
CURIE
XFrames
HLink
XHTML+MathML+SVG Profile
XHTML Modularization 1.0 Second Edition

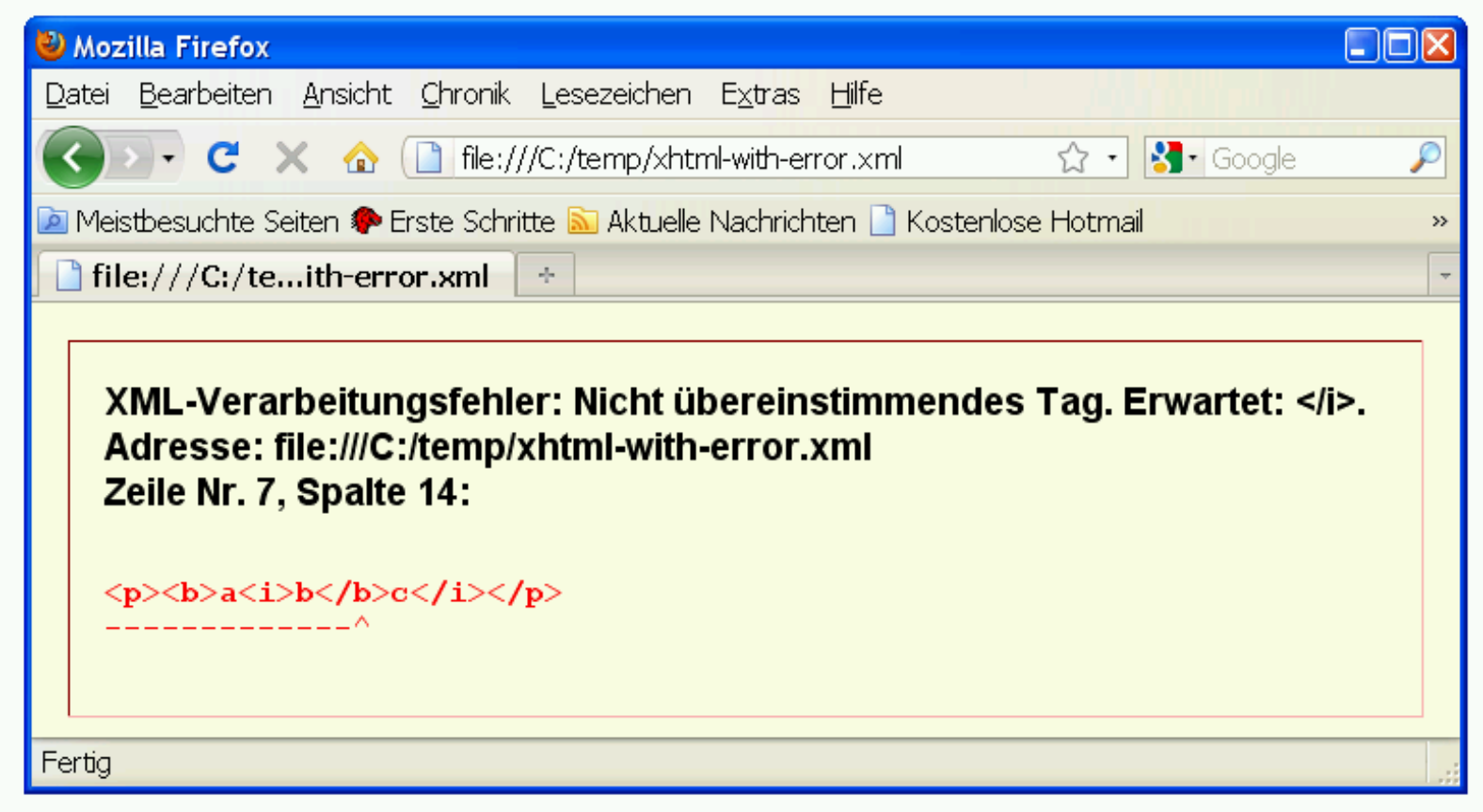
Hintergrund:
Das heutige Web

Just 4.13% of Web's Code is Valid

Vgl. „MAMA: What is the Web made of?“
http://dev.opera.com/articles/view/mama/

Hintergrund: Drakonische Fehlerbehandlung in XML

- abc im Browser:



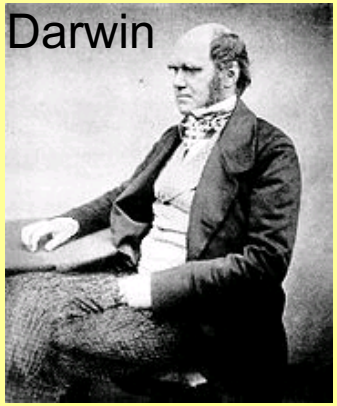
Just 4.13% of Web's Code is Valid

Just 4.13% of Web's Code is Valid

Nur XHTML wäre Revolution!

Che Guevara

Charles Darwin

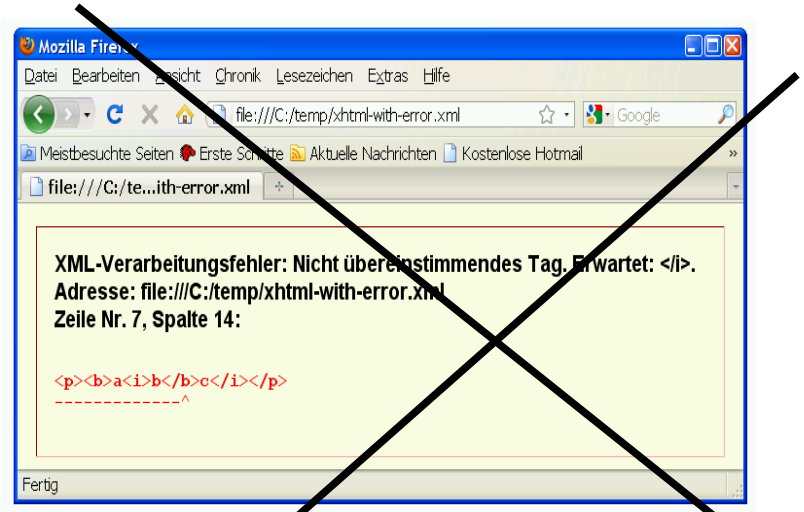


HTML5 ist Evolution

HTML5 Design Prinzipien

→ http://www.w3.org/TR/html-design-principles/

HTML5 Design Principle: „Support Existing Content“



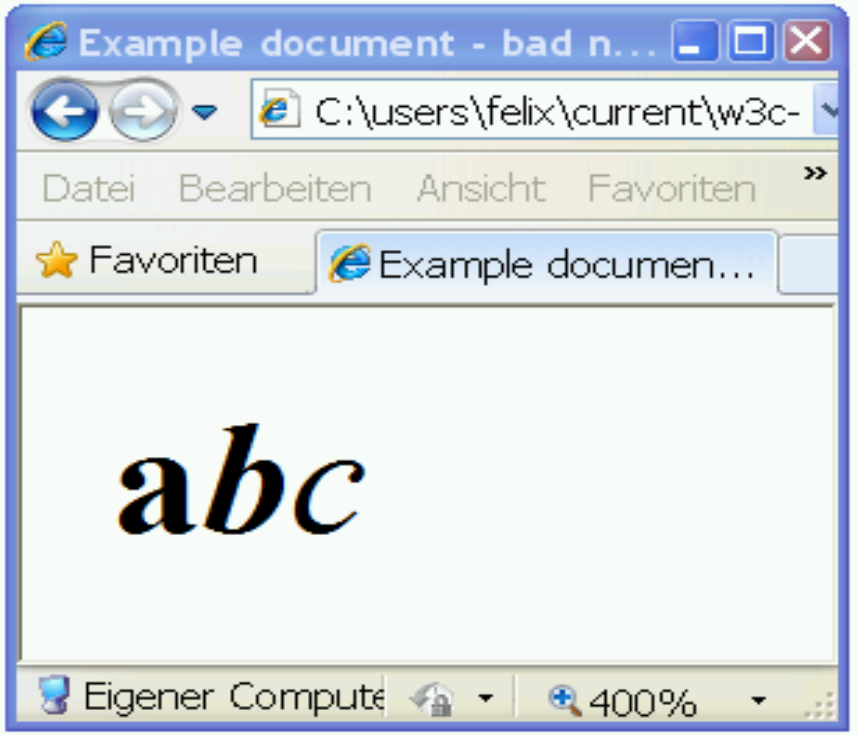
HTML5

HTML5 Design Principle: „Degrade gracefully“

```
<canvas ...>
<p>Sorry, but your browser does not support canvas :( </p>
</canvas>
```

HTML5 Design Principle: Interoperability: u.a. „Handle errors“

- abc



HTML5 Design Principle: „Dom Consistency“

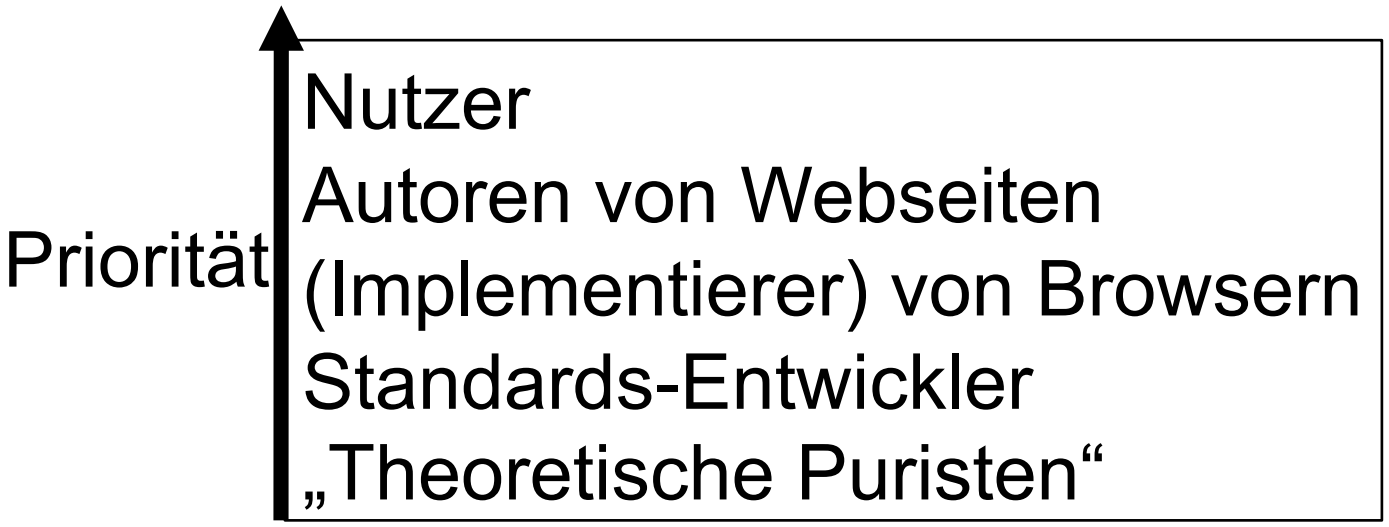
```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Example doc</title>
  </head>
  <body>
    <p>Example paragraph</p>
  </body>
</html>
```

```
<?xml version="1.0"encoding="UTF-8"?>
<html
xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Example document</title>
  </head>
  <body>
    <p>Example paragraph</p>
  </body>
</html>
```

Verschiedene Serialisierungen

Ein „Document Object Model“

HTML5 Design Principle: „Priority of Constituencies“



Grund für „Willful violations“ anderer Technologien

vgl. http://blog.whatwg.org/response-to-notes-on-html-5

Hat XML eine Zukunft im Web?

- Ja: im „tiefen Web“
  - Dateien in XML-Formaten: EAD, TEI, DocBook, ...
- Ungelöstes Problem: wie kommt der Informationsgehalt dieser Dokumente an der „Weboberfläche“ an?

Literatur und Tools

- Onlinequelle: TEIA-Lehrbuch zu XML
  - Vgl. http://www.teialehrbuch.de/Kostenlose-Kurse/XML/
  - Umfassende Einführung
- XML in der Praxis
  - Vgl. http://www.linkwerk.com/pub/xmlidp/2000/
  - Übersicht zu den wichtigsten Themen
- Editor oXygen
  - Vgl. http://www.oxygenxml.com/
  - Enthält TEI, EAD etc. schon vorkonfiguriert

Literatur und Tools

- Editor EditiX (freie Version),
  - Vgl. http://www.free.editix.com/
- Saxon XSLT/XQuery Prozessor
  - Vgl. http://saxon.sourceforge.net/

Einführung in XML

Felix Sasaki

Version: November 2016