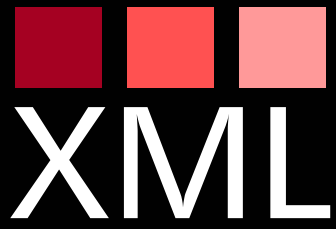


Chaos Seminar
10. Mai 2004

Frank Kargl
frank.kargl@ulm.ccc.de

Überblick

- Historie
- Markup Sprachen
- SGML
- XML
- XML-Technologien
- Bewertung

Heise Newsticker Suche

- XML: 406 Treffer
- TCP/IP: 116 Treffer
- Linux: 3493 Treffer

Aus dem Heise Forum …

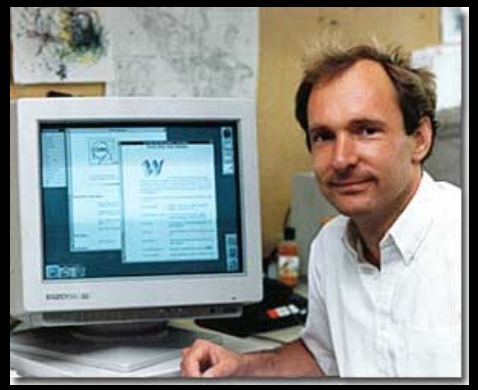
Noch eine Sprache? Wozu HTML hat sich bewährt.
AOL Fan (11. Februar 2004 14:08)

Ich verstehe ja, daß die vom w3c für Fortschritt sorgen wollen.
Aber braucht man denn wirklich noch eine Sprache, wenn sich die alte bewährt hat.
Genauso war es doch mit XML. Ich habe noch nie eine Seite gesehen, die in XML gecodet wurde.
Auch wenn es einigen Fortschrittstrunkenen nicht passt. HTML ist und bleibt die Sprache für das Internet Nummer 1.
Daran werden auch tausend Spezifikationen nichts dran ändern, egal wie gut sie nun gemeint sind. Solange die Browserhersteller nicht mitziehen und dieses WOL implementieren passiert da gar nichts.
Manchmal muss man einfach auch Pragmatiker bleiben.

Grüße

WorldWideWeb am CERN

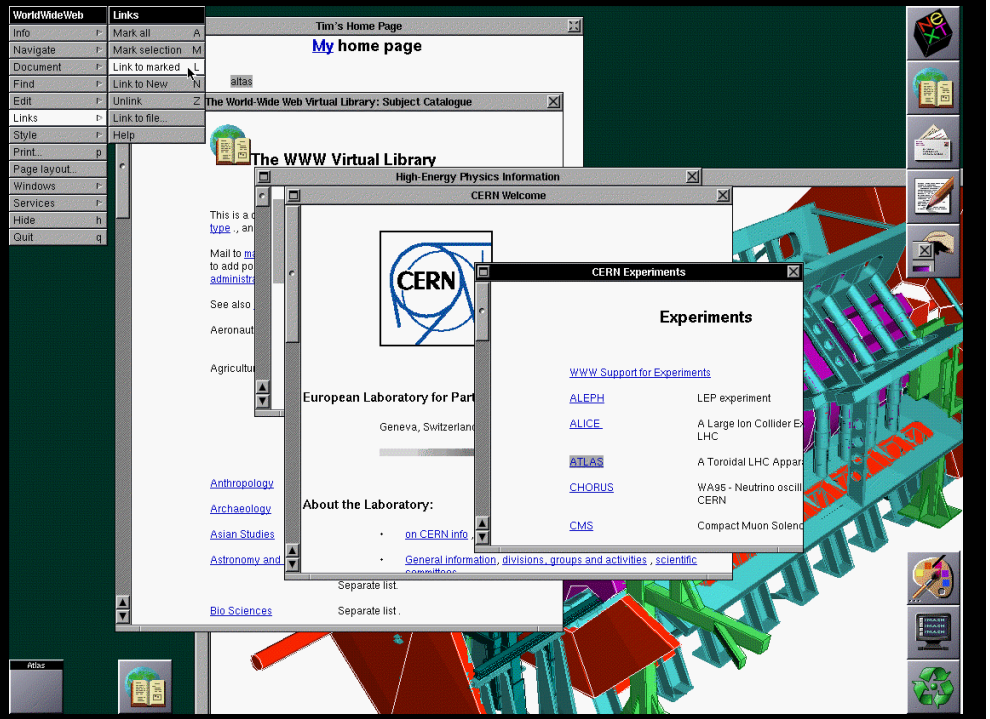
- 1989 geplant als Information Infrastructure für Hochenergiephysik
  - Integration verschiedenster bestehender Informationsquellen
  - "Information Management: A Proposal" von Tim Berners-Lee am CERN verteilt.



n ì ÉääÉW=**http://www.w3.org/History.html**

WorldWideWeb am CERN

- 1990 Kauf einer NeXTCube zur Realisierung des WorldWideWeb
- August 1991 Ankündigung in alt.hypertext, verfügbar per FTP
- Dezember 1991 Hypertext'91



n ì ÉääÉW=**http://www.w3.org/History.html**

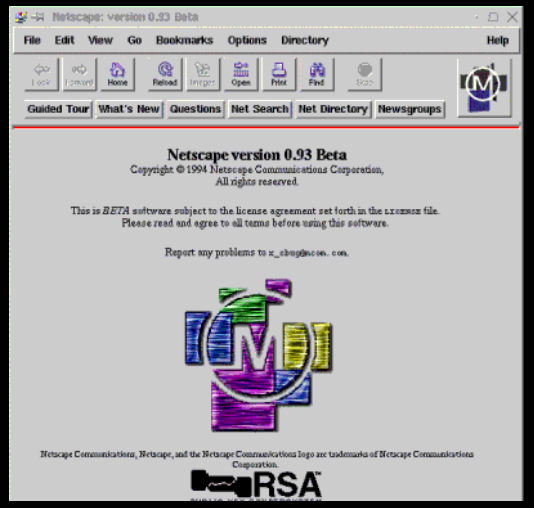
WorldWideWeb am CERN

- Webpage info.cern.ch von 1992:
  http://www.w3.org/History/19921103-hypertext/hypertext/WWW/TheProject.html
- April 1993: CERN gibt Webtechnologie frei
- Anfang 1993:
  Erste Versionen des
  Browsers „X Mosaic“
  http://www.webhistory.org/www.lists/www-talk.1993q1/0099.html

n ì ÉääÉW=**http://www.w3.org/History.html**

WorldWideWeb am CERN

- März 1994:
  Marc Andreessen und Jim Clark
  gründen „Mosaic Communications Group“
  (später Netscape)
- Oktober 1994:
  Gründung
  World Wide Web
  Consortium



n ì ÉääÉW=**http://www.w3.org/History.html**

hçãéçåÉåíÉå-ÉáåÉë-îÉêÍÉäíÉë-açâìãÉåíÉëóëíÉã

- Wie werden Dokumente kodiert?
  - Inhalt
  - Semantik
  - Darstellung
- Wie werden Dokumente identifiziert?
  - Wo befinden sich die Daten?
  - Wie kann auf die Daten zugegriffen werden?
- Wie werden die Dokumente zum Benutzer transportiert?

Ansatz im Web

- Dokumentformat
  - Hypertext Markup Language, HTML
    - Document Type Definition (DTD)
      Standardized General Markup Language (SGML)
- Identifikationsschema
  - Uniform Resource Locator, URL
    - Lokation als DNS Name
    - Protokoll als Schema
- Transferprotokoll
  - Hypertext Transfer Protocol, HTTP
    - ASCII-kodiertes Request-Reply Protokoll über TCP/IP

HTML - Entwurfsziele

- Einfach
  - Auch Nicht-Computer-Experten sollen Autoren sein
- Anwendungsübergreifend
  - Vielzahl von Anwendungen soll möglich sein
- Plattformunabhängig
  - Fokus auf Inhalt, nicht auf Darstellung
- SGML als Definitionsmittel
  - HTML als DTD beschreiben (tatsächlich erst ab HTML 2.0)

Geschichte

SGML
1986
Ur-HTML
1989
1990
Mosaic
HTML+
"HTML 1"
1991
1992
Mosaic
HTML+
"HTML 1"
1993
Netscape
HTML 2
1994
Microsoft
1995
HTML 3.0
1996
HTML 3.2
1997
XML 1.0
HTML 4.0
1998
XSL
XLink
XPointer
XQuery
XMLSchema
WML
MathML
HTML 4.01
1999
XML 1.0 (2nd ed.)
XHTML 1.0
2000
XHTML 1.1
2001
2002
XML 1.1 cand. rec.
XHTML 2.0 Draft
2003

Markup Sprachen

- Ursprung: Computerized Typesetting
  - Typesetting-Makros eingebettet in ASCII
- Presentational Markup
  - Anweisungen zu Formatierung und Layout
    - z.B. Schriftart, -größe, -stil, Ausrichtung, Randabstände usw.
  - Beispiele:
    - TeX
    - nroff
- Semantisches Markup
  - Autoren versehen Inhalte mit vereinbarten Annotationen
    Verleger entscheiden über Layout
  - Semantisches Markup unterstützt Suche in Dokumenten
    - Markup für Maschinen vs. Markup für Menschen

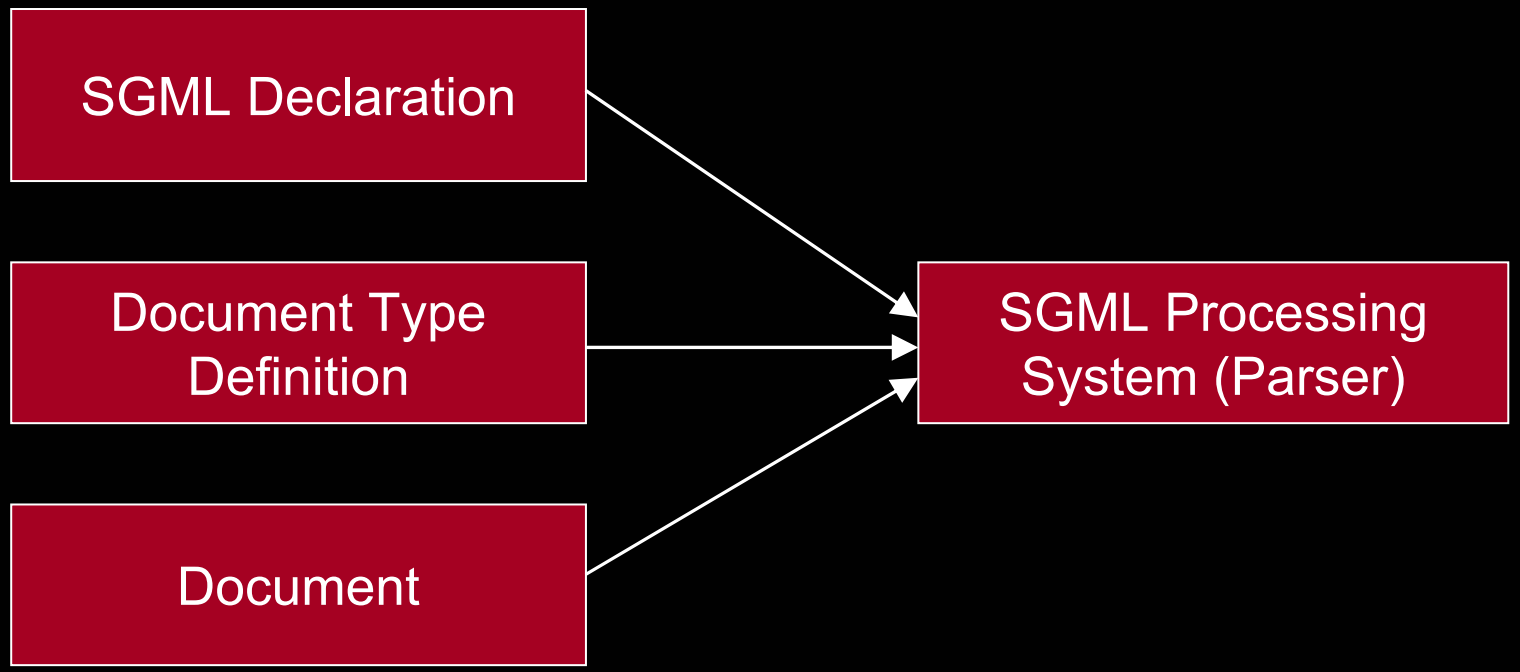
SGML

- Standard Generalized Markup Language
  - ISO-Standard [ISO/IS 8879, 1986]
  - Entstanden aus GML, IBM 1969
    (nach Goldfarb, Mosher, Lorie)
  - Trennung von Inhalt und Präsentation
- Dilemma konkreter Markup-Sprachen:
  - Welche Tag-Menge vorgeben?
- Generalized Markup
  - Beschreibe Dokumente in drei Teilen
    - SGML Declaration:
      Abbildung der abstrakten SGML Syntax auf konkrete Syntax
    - Document Type Definition, DTD:
      Definition der Tags und ihrer Bedeutung
    - Eigentliches Dokument mit Markup

SGML Konzepte

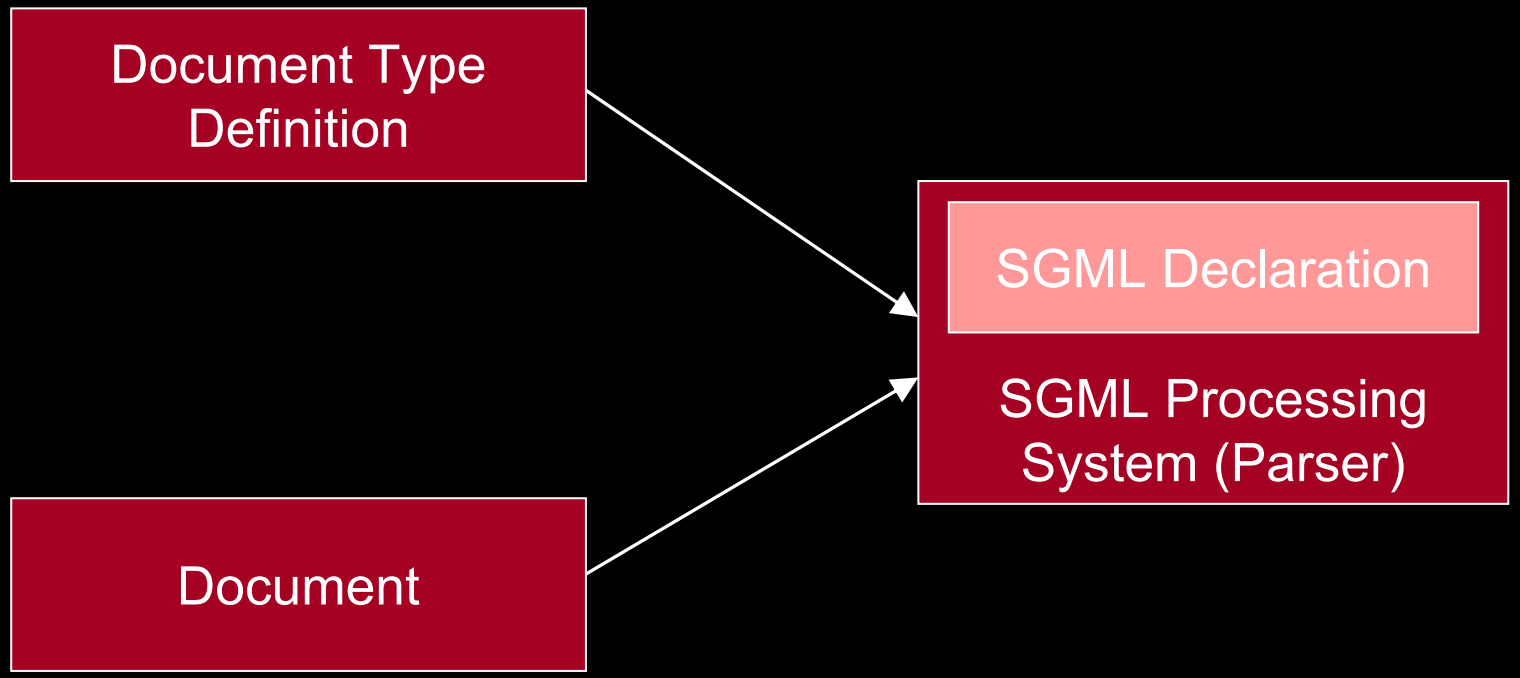
- Deskriptives statt prozedurales Markup
  - “this is an X” statt “do X here”
- Entity
  - Sammlung von Zeichen die als Einheit referenziert werden können
    „collection of characters that can be referenced as unit“
- Element
  - Komponente in der hierarchischen Struktur die durch die DTD definiert ist
    „component of the hierarchical structure defined by a document type definition“
- SGML verwendet Tags zur Beschreibung der Element Structure
- Entities werden als Bezugspunkte definiert
- Dokumentenklasse
  - Menge aller von einer DTD beschriebenen Dokumente

Generelles SGML Processing



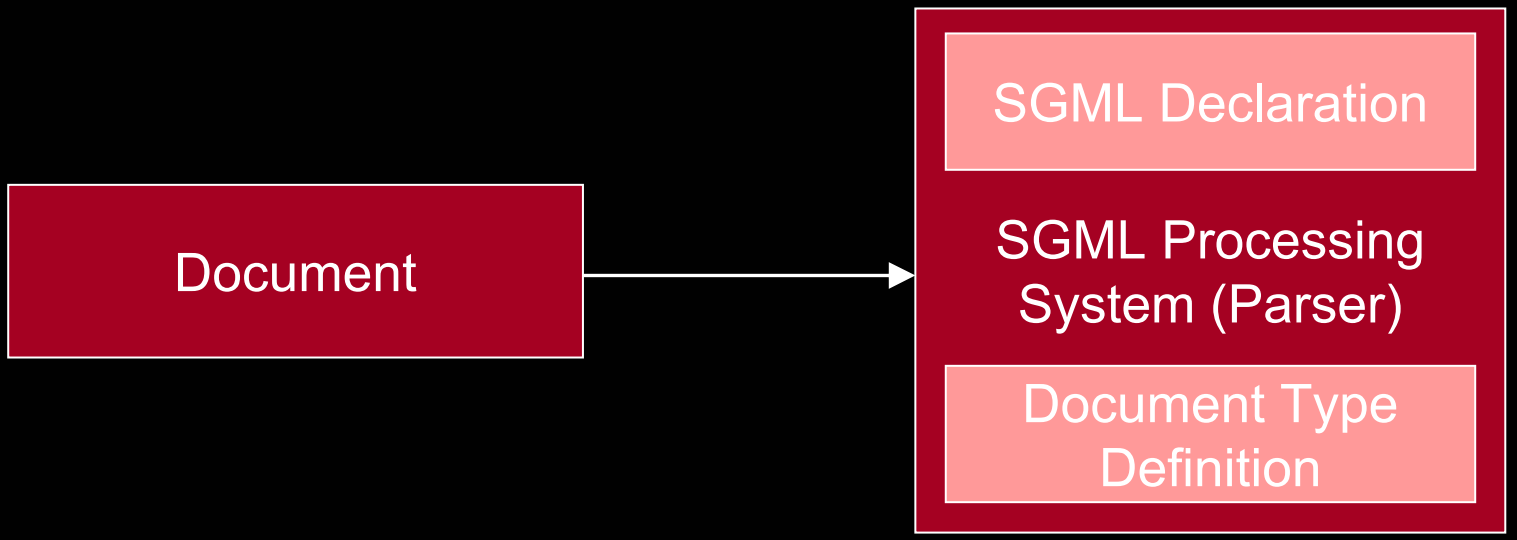
Spezielles SGML Processing I

- SGML Syntax festgelegt
  - z.B. bei XML (Extensible Markup Language) (SGML-Profil)



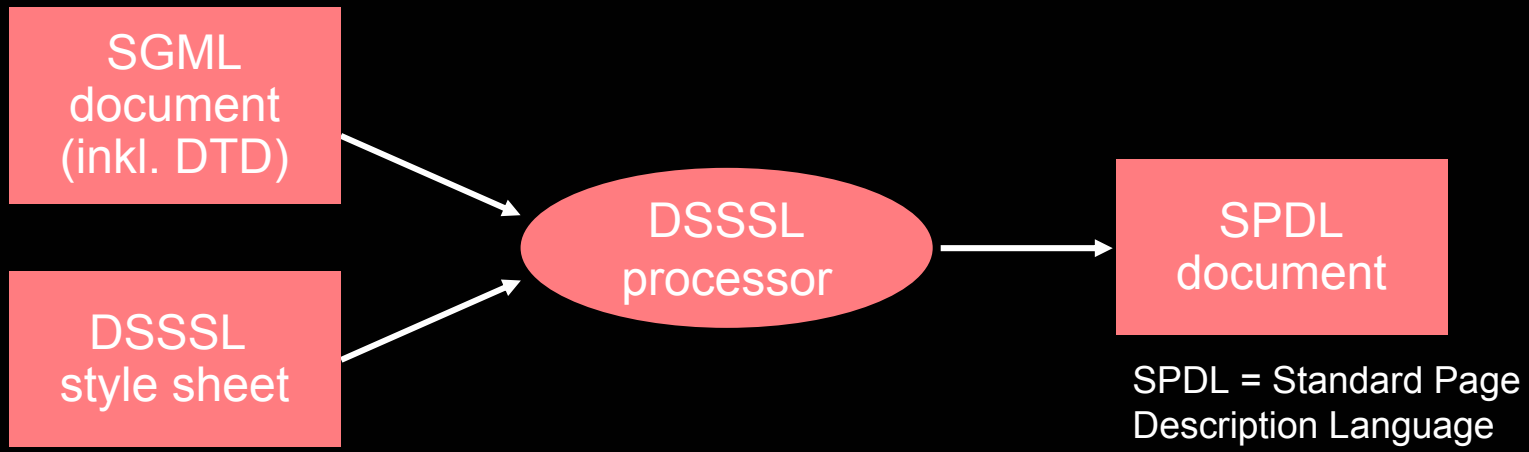
Spezielles SGML Processing II

- Document Type Definition und SGML Syntax festgelegt
  - z.B. bei HTML (Hypertext Markup Language)
  - Nur eine Dokumentenklasse



Formatierung von Dokumenten I

- HTML
  - Automatisch formatierbar
  - Aber eingeschränkt auf bestimmten Dokumenttyp mit wenigen Tags
- SGML
  - beliebige Dokumenttypen definierbar, aber nur die Semantik
  - Formatierung durch *Document Style Specification and Semantics Language*, DSSSL



SPDL = Standard Page Description Language

Formatierung von Dokumenten II

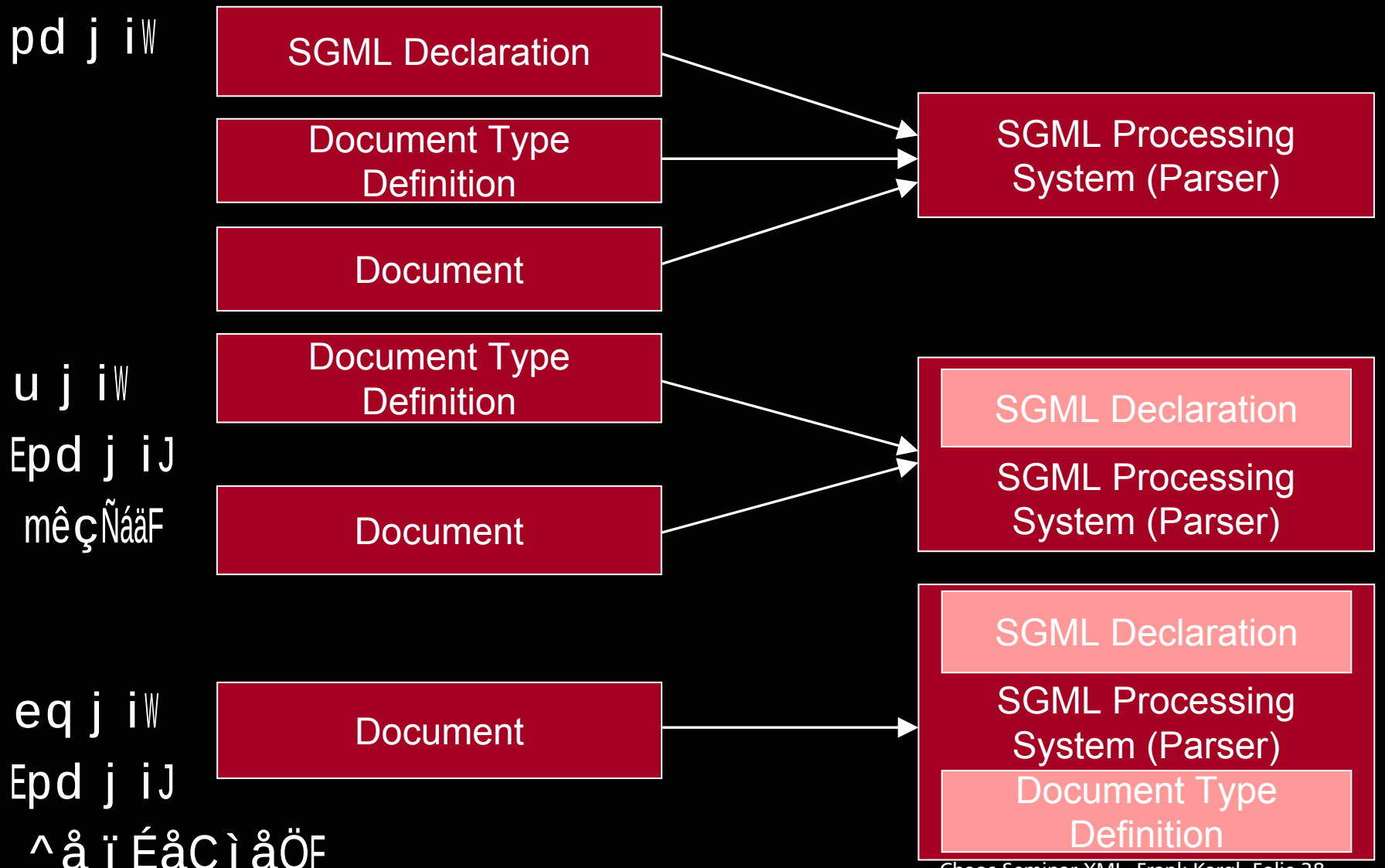
- DSSSL
  Document Style Specification and Semantics Language
  [ISO/IS 10179, 1996]
  - Umformen von SGML Dokumenten eines Dokumenttyps in einen anderen
  - Umformen in ein Ausgabeformat
    - SPDL: Standard Page Description Language (Postscript-ähnlich)

- Transformation von SGML-Dokumenten
  - DSSSL baut mit Hilfe der DTD die Baumstruktur des SGML Dokuments auf
  - Standard Tree Formatting Process, STFP, produziert Ausgabe gemäß Style Sheet

Warum XML?

- HTML sehr erfolgreich
- Aber: HTML sehr eingeschränkt
  - Endliche Menge von Tags
  - Kein spezifischer, semantischer Markup (im Bezug auf Dokumentinhalt)
- Leistungsfähiger: SGML
- Aber: sehr kompliziert
  - Viele Freiheit in Bezug auf Syntax
  - Schwer automatisiert zu parsen
  - Programme implementieren nur Teilmengen ➔ Inkompatibiliät
- XML als Mittelweg

Vergleich


pd j iW
SGML Declaration
Document Type Definition
Document
SGML Processing System (Parser)
u j iW
Epd j iJ
mêçÑáäF
Document Type Definition
Document
SGML Declaration
SGML Processing System (Parser)
eq j iW
Epd j iJ
^å ï ÉåÇ ì åÖF
Document
SGML Declaration
SGML Processing System (Parser)
Document Type Definition

Entwicklung

- 1996 Bosak, Bray, et.al. beginnen mit Arbeit an SGML-Lite
- 1998 XML 1.0
- Vielzahl weiterer Standards
  - Namensräume
  - Stylesheets
  - Xlink, XPath, XPointer
  - HTML-DOM wird auf XML erweitert
  - APIs: SAX, JAXP, ...

Anwendungsgebiete

Textorientierte Anwendungen

- Erweiterung von HTML
- Narrativer, textorientierter Inhalt
- Wird um Markup angereichert.
- Beispiele: TEI, Docbook, XHTML

Datenorientierte Anwendungen

- Verwendung von XML zur Strukturierung von Daten.
- Z.B. Datenbanken, Tabellenkalkulation usw.
- Problem: keine Datentypen in DTD
- Beispiele: SOAP, SVG, ebXML

XML Anwendungen

- Große Vielzahl standardisierter XML-Anwendungen
  - Molecular Dynamics Language (MoDL)
  - Chemical Markup Language
  - Mathematical Markup Language
  - Music Markup Language
  - ICE (Information and Content Exchange Protocol)
  - Resource Description Framework (RDF)
  - Weather Observation Markup Format (OMF)
  - Extensible Logfile Format (XLF)
  - Extensible Mail Transport Protocol (XMTP)
  - Personalized Information Description Language (PIDL)
  - XHTML
  - Channel Definition Format (CDF)
  - Open Software Description Format (OSD)
  - Scalable Vector Graphics (SVG)
  - Rich Site Summary (RSS)
  - Microsoft Office 2003 XML Formate
  - SOAP/XML-RPC
  - uvm.

XHTML 1.0

- Neuformulierung von HTML 4 in XML
- Semantik der Elemente und Attribute wie HTML 4
- Vorteile von XHTML
  - XHTML Dokumente sind XML konform
  - XHTML erlaubt die Verwendung des HTML oder XML Document Object Models
  - Verarbeitung der Dokumente (z.B. im Browser, Editor) wird einfacher
  - HTML 4.0 Browser sind in der Lage diese Dokumente anzuzeigen
  - Auf dieser Basis wird es einfacher, neue Elemente zu definieren, um z.B. semantisch reichere Dokumente zu schreiben

Gültiges XHTML

- Kriterien:
  - Entspricht einer gültigen XHTML DTD
    (1.0: strict, transitional, frameset, 1.1: xhtml11 + Module)
  - Wurzelelement ist <html>
  - Die Wurzel verwendet den XHTML Namensraum
  - Das Dokument enthält eine DOCTYPE Deklaration
- Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
   <title>Virtual Library</title>
</head>
<body>
   <p>Moved to <a href="http://vlib.org/">vlib.org</a>.</p>
</body> </html>
```

Unterschiede von XHTML zu HTML

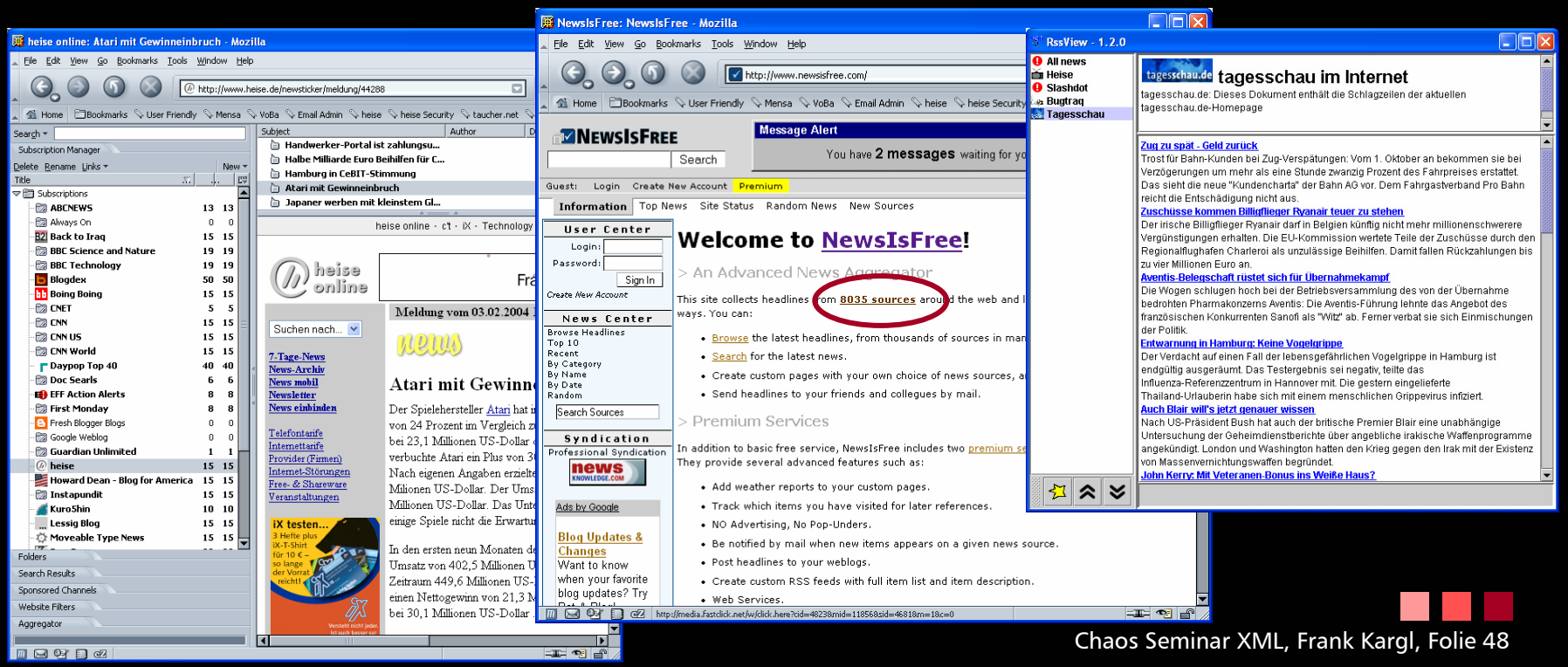
- „Well-Formed-Documents“
- Groß- und Kleinschreibung wird unterschieden
- Kein OMITTAG/SHORTTAG Feature bei XML:
  - End-Tags sind verpflichtend, oder <.. />
  - Attributwerte müssen in Anführungszeichen stehen
  - Attribut-Minimierung ist nicht erlaubt

RSS

- Unterschiedliche Bezeichnungen
  - Rich Site Summary (1997 urspr. Bez. von Netscape)
  - RDF Site Summary
  - Really Simple Syndication
- Unterschiedliche (teilw. inkompatible!) Versionen
- Begriffe
  - RSS Feed
  - RSS Aggregator
- Darstellung
  - Eigener Client
  - Integriert in Browser
  - Integriert in Website
- Siehe http://web.resource.org/rss/1.0/

RSS Beispiel

```
<item>
  <title>Earth Invaded</title>
  <link>http://news.example.com/2004/12/17/invasion</link>
  <description>The earth was attacked by an invasion fleet
  from halfway across the galaxy; luckily, a fatal
  miscalculation of scale resulted in the entire armada
  being eaten by a small dog.</description>
</item>
```



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns="http://purl.org/rss/1.0/" xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
   xmlns:content="http://purl.org/rss/1.0/modules/content/">
  <channel rdf:about="http://www.tagesschau.de/newsticker.rdf">
    <title>tagesschau im Internet</title>
    <link>http://www.tagesschau.de</link>
    <image rdf:resource="http://www.tagesschau.de/logo.gif" />
    <description>tagesschau.de: Dieses Dokument enthält die
     Schlagzeilen der aktuellen tagesschau.de-Homepage</description>
    <dc:publisher>tagesschau.de</dc:publisher>
    <sy:updatePeriod>hourly</sy:updatePeriod>
    <items><rdf:Seq>
        <rdf:li resource="URI1" />
        <rdf:li resource="URI2" />
    </rdf:Seq></items>
  </channel>
  <image rdf:about="http://www.tagesschau.de/logo.gif">
    <title>tagesschau.de</title>
    <link>http://www.tagesschau.de</link>
    <url>http://www.tagesschau.de/logo.gif</url>
  </image>
  <item rdf:about="URI1">
    <dc:date>2004-02-03T15:30:00+01:00</dc:date>
    <dc:format>text/html</dc:format>
    <title>Zug zu spät - Geld zurück</title>
    <link>URI1</link>
    <description>
Trost für Bahn-Kunden bei Zug-Verspätungen: Vom 1. Oktober an  bekommen sie bei
   Verzögerungen um mehr als eine Stunde zwanzig Prozent des Fahrpreises erstattet.
   Das sieht die neue "Kundencharta" der Bahn AG vor. Dem Fahrgastverband Pro Bahn
   reicht die Entschädigung nicht aus.
    </description>
  </item>
  ...
</rdf:RDF>
```

Wohlgeformte XML Dokumente

Regeln für wohlgeformte (well-formed) XML-Dokumente:

1. Jedes Start Tag hat zugehöriges End Tag
   (Abkürzende Schreibweise: <tag />)
2. Elemente dürfen geschachtelt sein, sich aber nicht überlappen
3. Es muss genau ein Wurzelelement geben
4. Attributwerte müssen in Anführungszeichen stehen
5. Ein Element darf nicht zwei Attribute mit dem gleichen Namen besitzen
6. Kommentare und Verarbeitungsanweisungen dürfen nicht innerhalb von Tags stehen.
7. In den Zeichendaten eines Elements oder Attributs dürfen keine geschützen < oder & Zeichen stehen.

Unterscheide:
wohlgeformt und valide (einer DTD entsprechend)

Aufbau XML Dokument

XML Deklaration

encoding: im Dokument verwendeter Zeichensatz (optional). default: utf-8

standalone: Zugriff auf externe DTD?

Dokumenttyp Deklaration (siehe später)

Dokumenten-Körper

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<!DOCTYPE person SYSTEM "http://www.da.de/dtd/person.dtd">
<person>
  <name>
    <vorname>Alan</vorname>
    <nachname>Turing</nachname>
  </name>
  <beruf>Informatiker</beruf>
  <beruf>Mathematiker</beruf>
  <beruf>Kryptograph</beruf>
</person>
```

XML Technologien

- Referenzierung
  - XPath
  - XLink
  - XPointer
- Schemabeschreibungen
  - DTDs
  - XML-Schema
- Transformation
  - XSLT
  - XSL-FO
- Programmierung
  - DOM
  - SAX

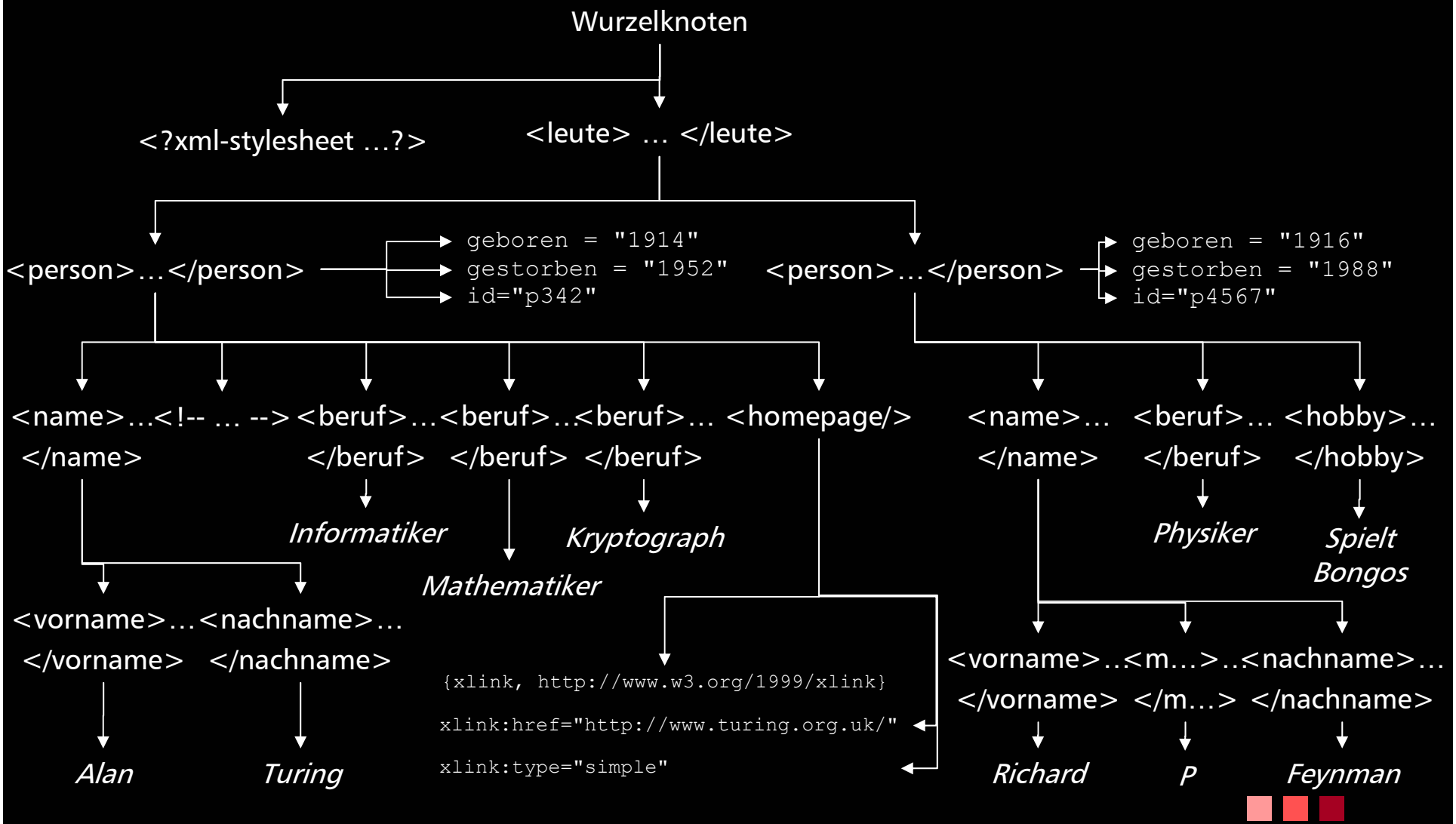
XPath

- Ziel:
  Zugriff/Referenzierung bestimmter Teile eines XML Dokuments
- Wird von verschiedenen anderen XML Standards eingesetzt
  (z.B. XPointer, XSLT, XML Schema, XForm)
- Idee:
  Auswahl eines/mehrerer Knoten(s) im XML Baum

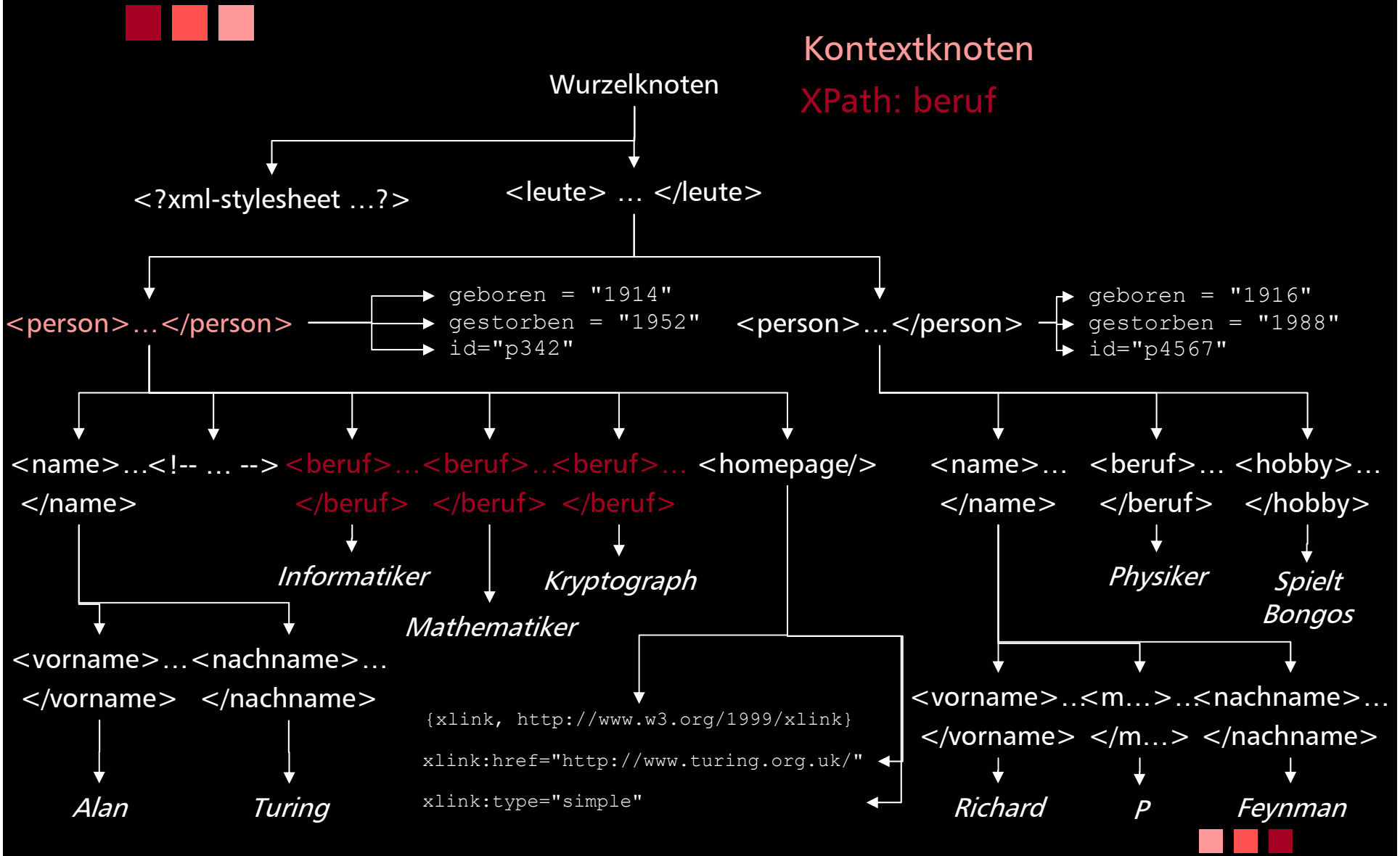
XPath Baumdarstellung

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="leute/xsl"?>
<!DOCTYPE leute [
 <!ATTLIST homepage xlink:type CDATA #FIXED "simple"
           xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink">
 <!ATTLIST person id ID #IMPLIED>
]>
<leute>
 <person geboren="1912" gestorben="1954" id="p342">
  <name>
   <vorname>Alan</vorname>
   <nachname>Turing</nachname>
  </name>
  <!-- Gab es das Wort Informatiker zu Turings Zeiten schon? -->
  <beruf>Informatiker</beruf>
  <beruf>Mathematiker</beruf>
  <beruf>Kryptograph</beruf>
  <homepage xlink:href="http://www.turing.org.uk/"/>
 </person>
 <person geboren="1918" gestorben="1988" id="p4567">
  <name>
   <vorname>Richard</vorname>
   <mittelinitial>&#x4D;</mittelinitial>
   <nachname>Feynman</nachname>
  </name>
  <beruf>Physiker</beruf>
  <hobby>Spielt Bongos</hobby>
 </person>
</leute>
```

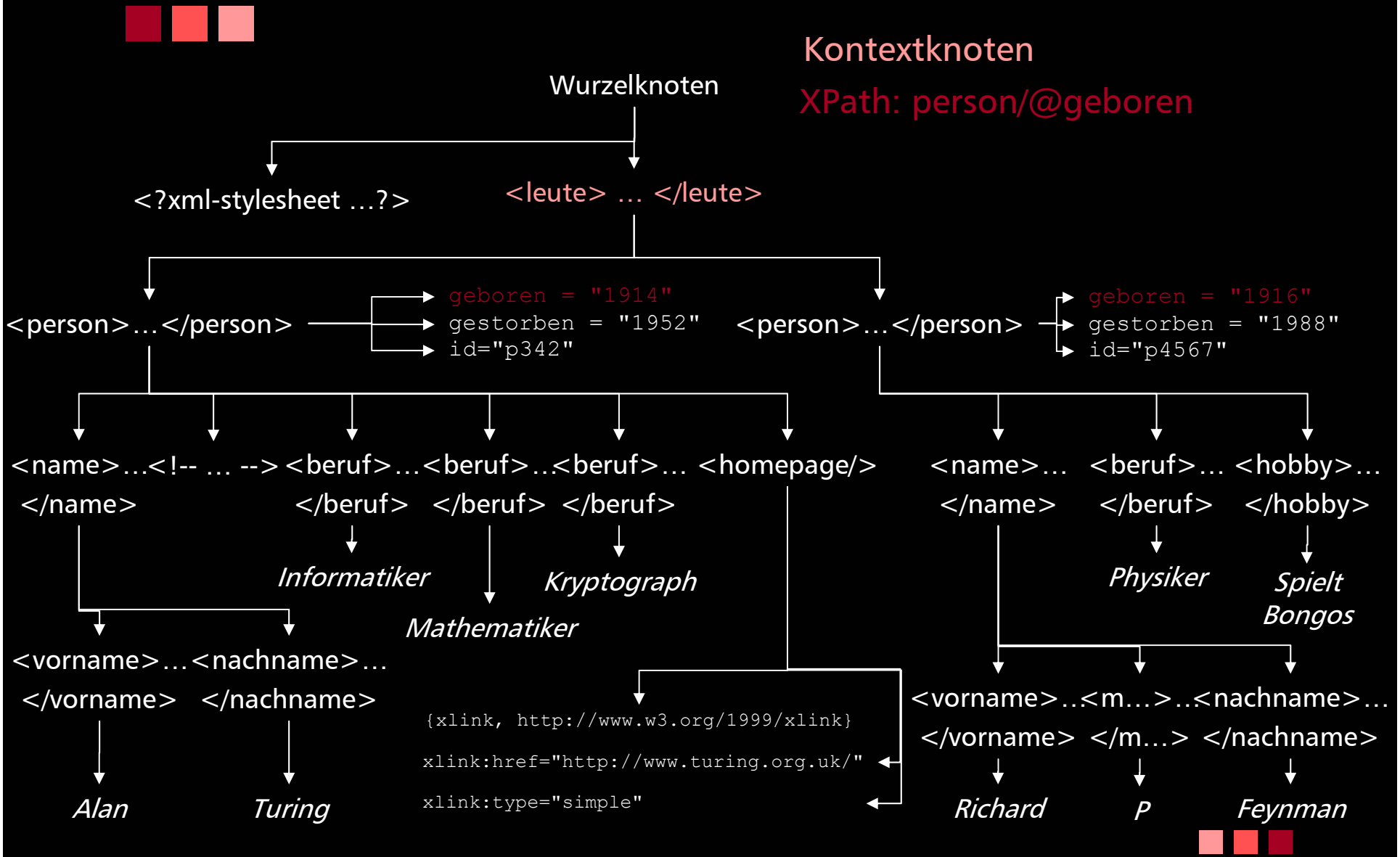
XPath Baumdarstellung



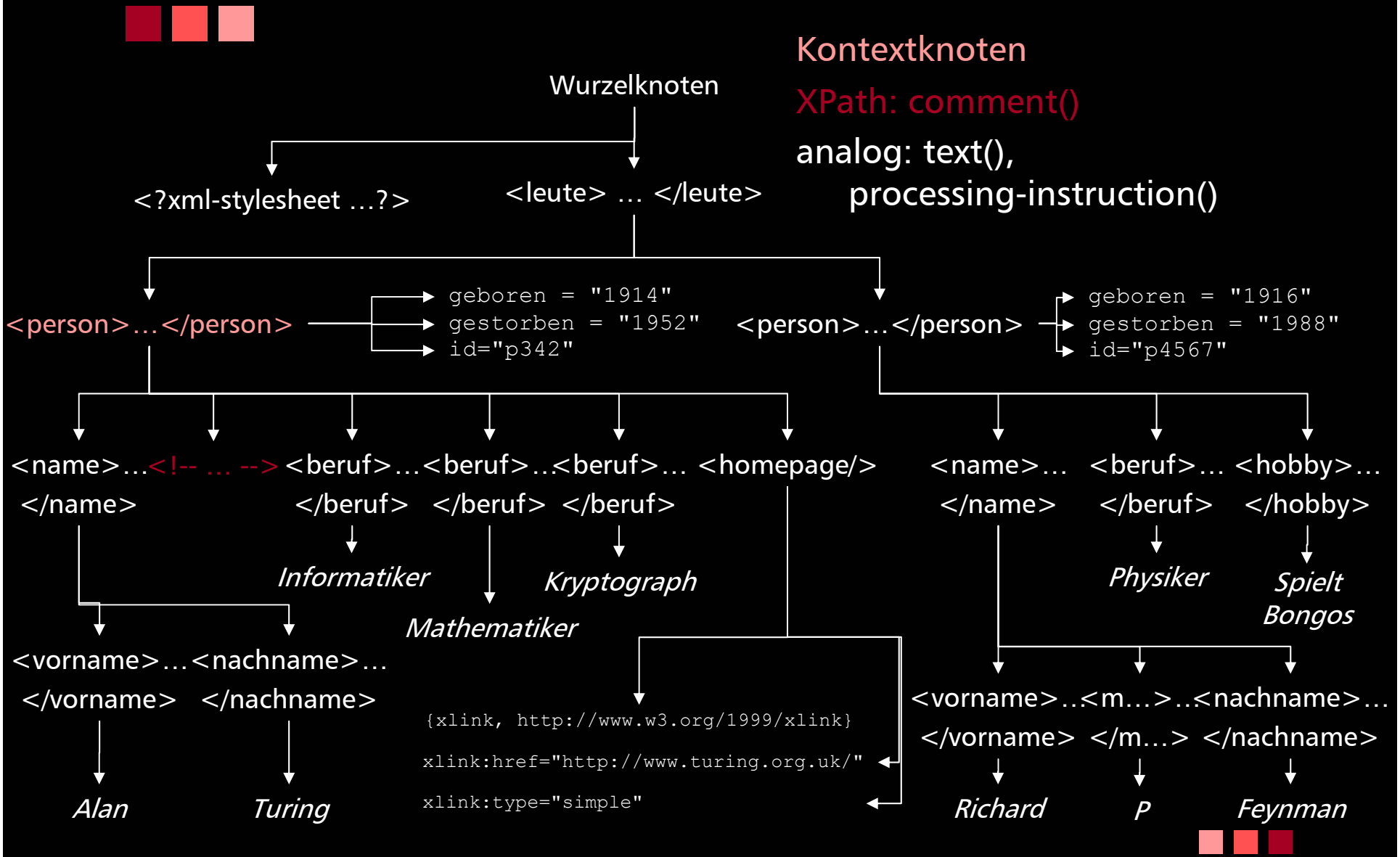
Lokalisierung Kindelemente



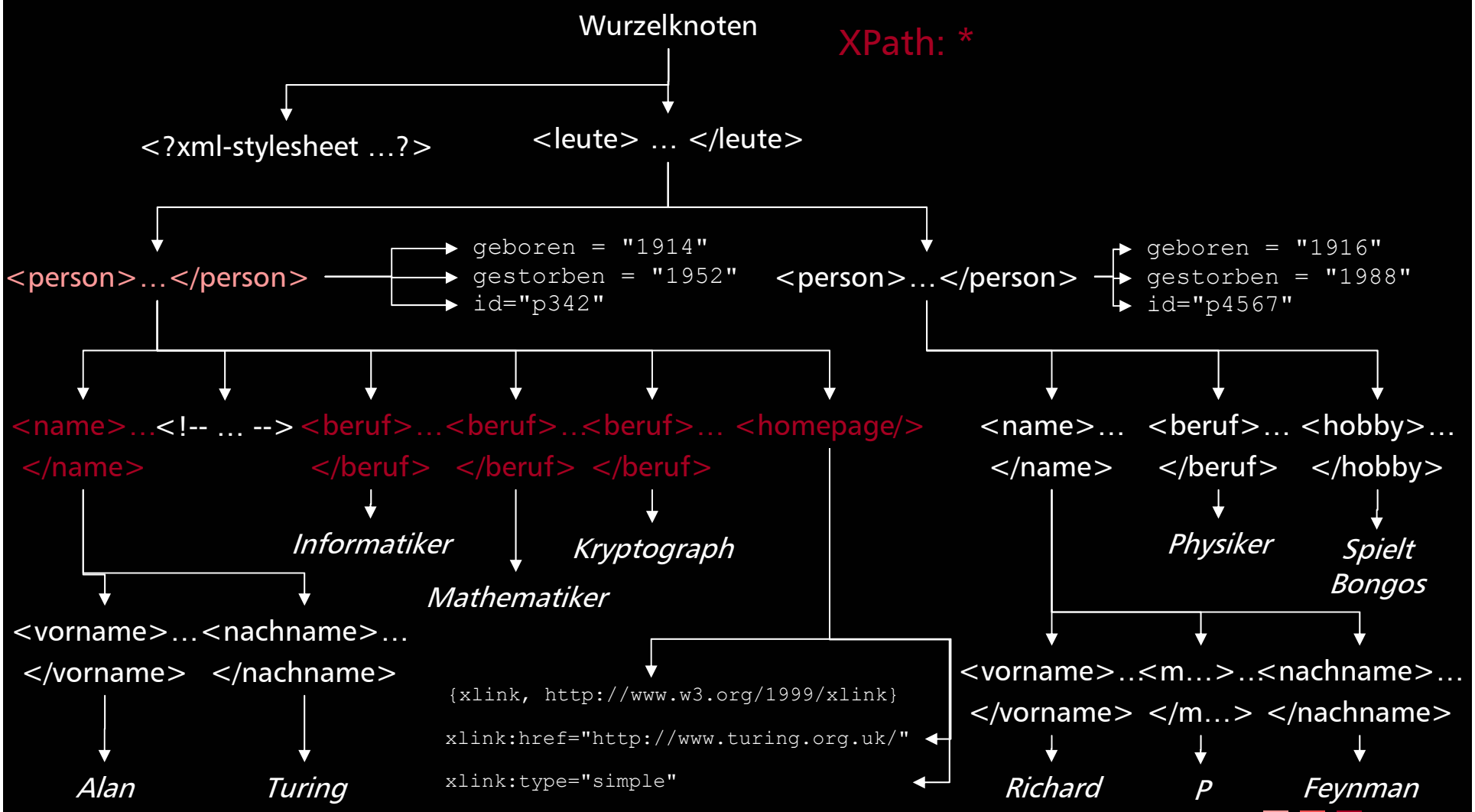
Lokalisierung Attribute



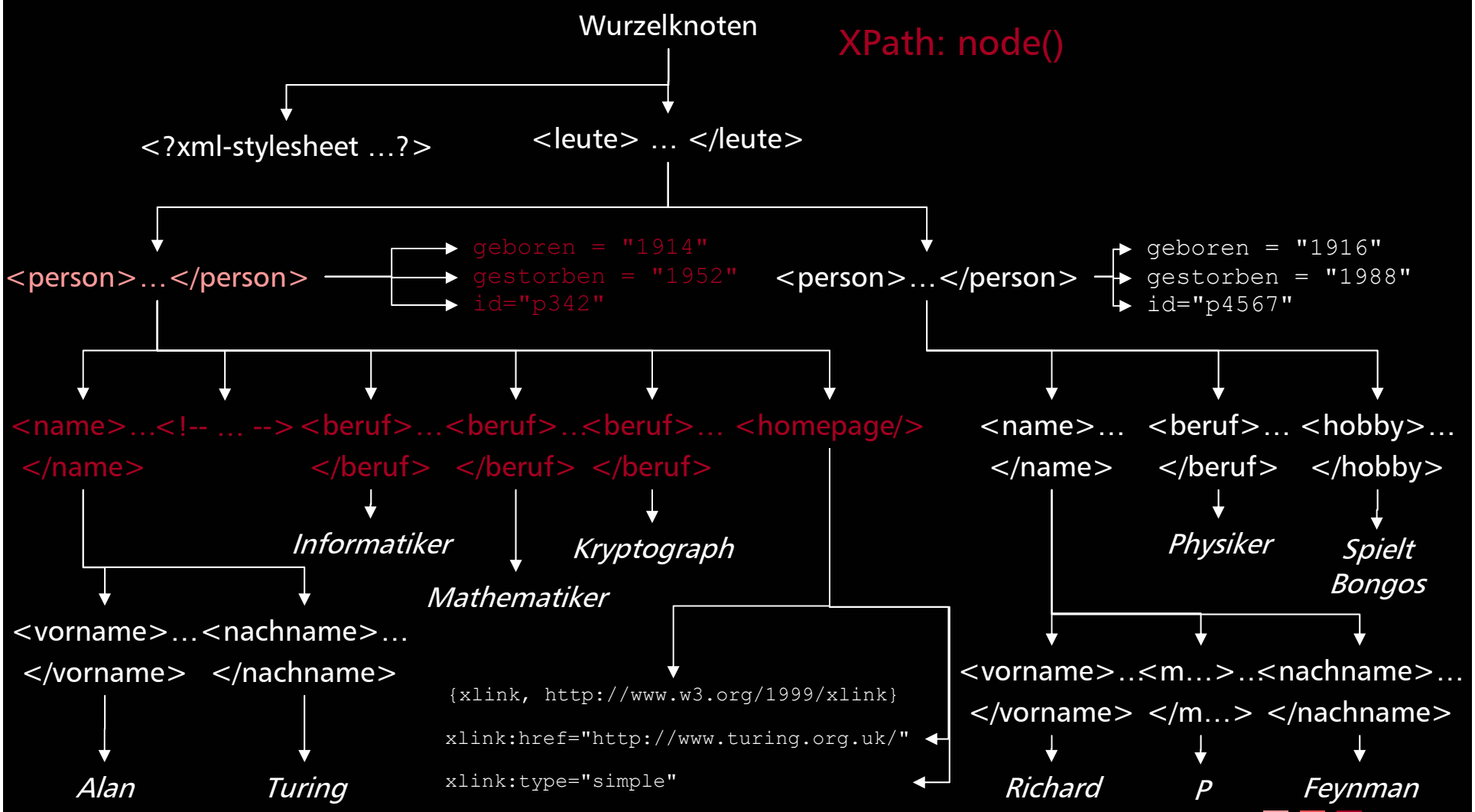
Kommentare, Textknoten, PIs



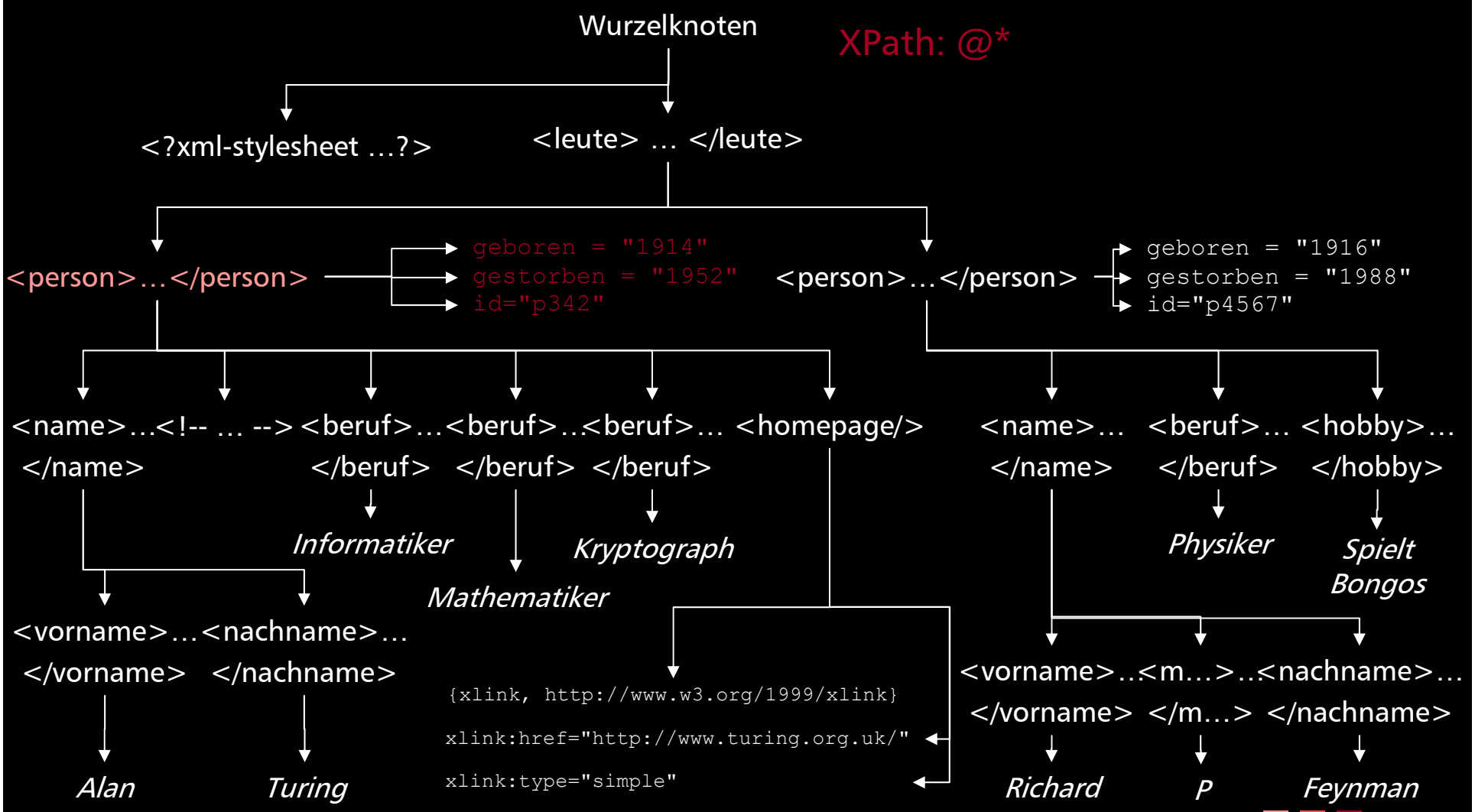
Wildcards


Kontextknoten
XPath: *
Wurzelknoten
<?xml-stylesheet …?>
<leute> … </leute>
<person>…</person>
geboren = "1914"
gestorben = "1952"
id="p342"
<person>…</person>
geboren = "1916"
gestorben = "1988"
id="p4567"
<name>…</name>
<!-- … -->
<beruf>…</beruf>
<beruf>…</beruf>
<beruf>…</beruf>
<homepage/>
<name>…</name>
<beruf>…</beruf>
<hobby>…</hobby>
Informatiker
Mathematiker
Kryptograph
Physiker
Spielt Bongos
<vorname>…</vorname>
<nachname>…</nachname>
{xlink, http://www.w3.org/1999/xlink}
xlink:href="http://www.turing.org.uk/"
xlink:type="simple"
<vorname>…</vorname>
<m…>…</m…>
<nachname>…</nachname>
Alan
Turing
Richard
P
Feynman

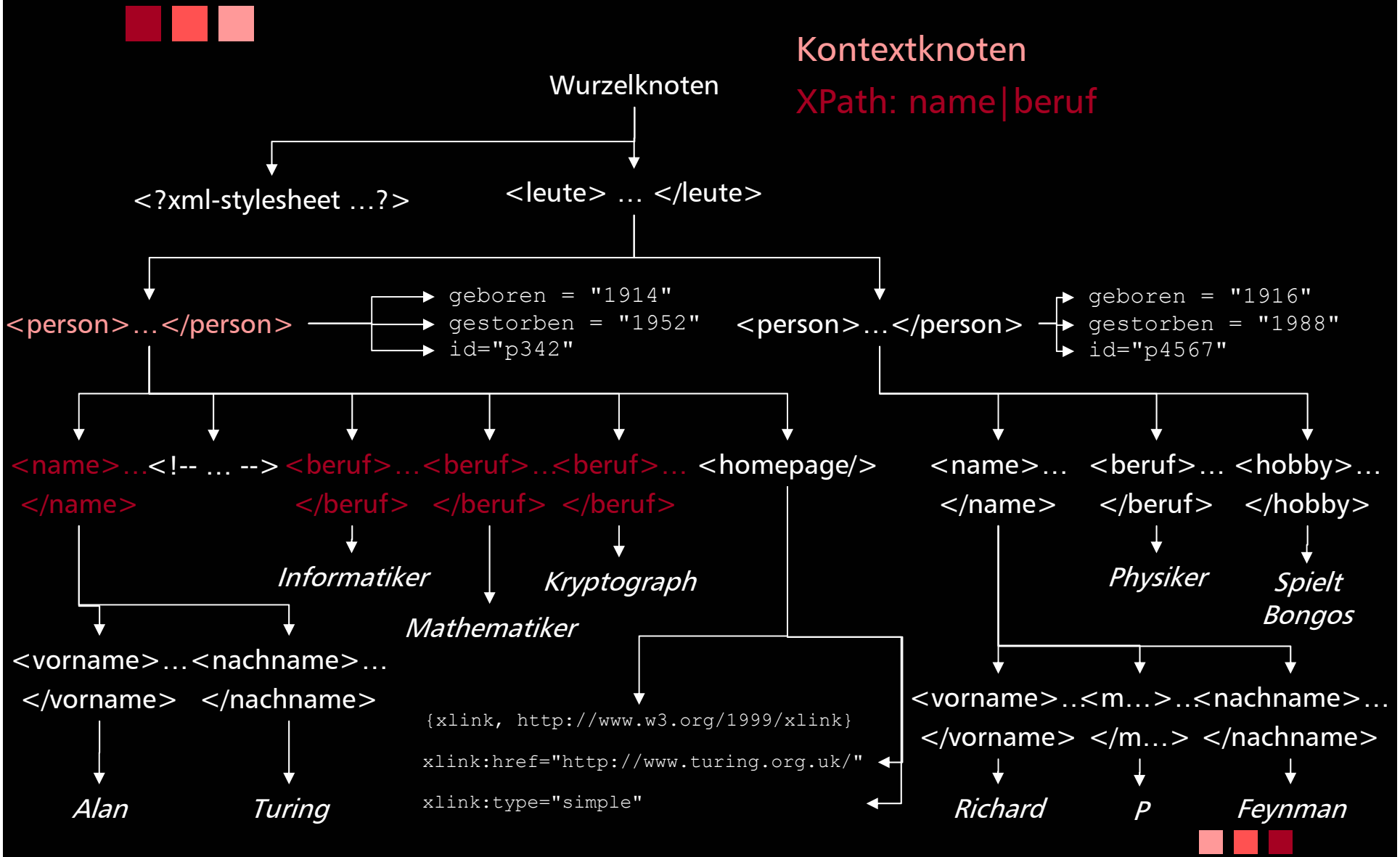
Wildcards


Kontextknoten
XPath: node()
Wurzelknoten
<?xml-stylesheet …?>
<leute> … </leute>
<person>…</person>
geboren = "1914"
gestorben = "1952"
id="p342"
<person>…</person>
geboren = "1916"
gestorben = "1988"
id="p4567"
<name>…<!-- … --></name>
<beruf>…</beruf>
<beruf>…</beruf>
<beruf>…</beruf>
<homepage/>
<name>…</name>
<beruf>…</beruf>
<hobby>…</hobby>
Informatiker
Mathematiker
Kryptograph
Physiker
Spielt Bongos
<vorname>…</vorname>
<nachname>…</nachname>
{xlink, http://www.w3.org/1999/xlink}
xlink:href="http://www.turing.org.uk/"
xlink:type="simple"
<vorname>…</vorname>
<m…>…</m…>
<nachname>…</nachname>
Alan
Turing
Richard
P
Feynman

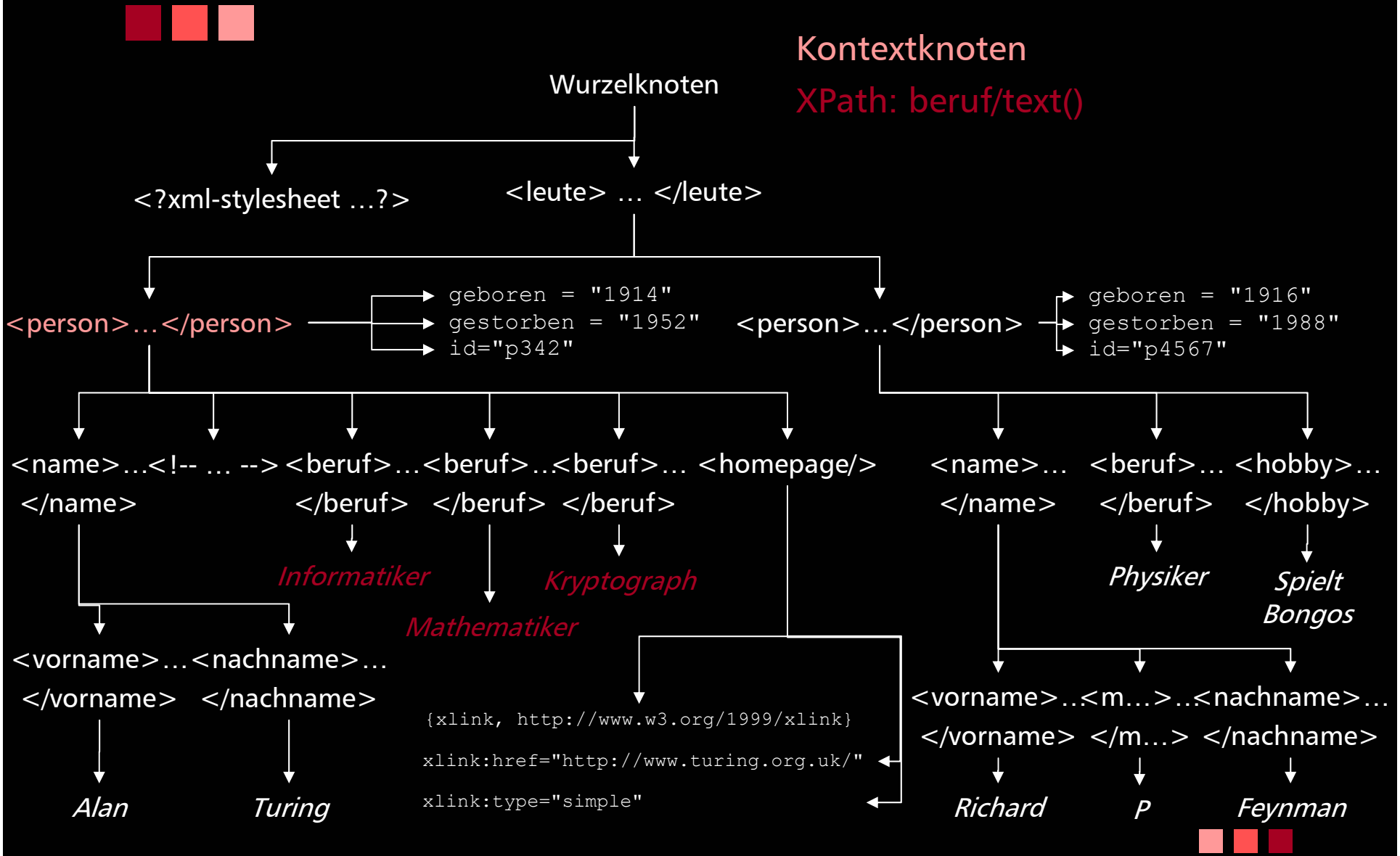
Wildcards


Kontextknoten
XPath: @*
Wurzelknoten
<?xml-stylesheet …?>
<leute> … </leute>
<person>…</person>
geboren = "1914"
gestorben = "1952"
id="p342"
<person>…</person>
geboren = "1916"
gestorben = "1988"
id="p4567"
<name>…<!-- … --> </name>
<beruf>… </beruf>
<beruf>… </beruf>
<beruf>… </beruf>
<homepage/>
<name>… </name>
<beruf>… </beruf>
<hobby>… </hobby>
Informatiker
Kryptograph
Mathematiker
Physiker
Spielt Bongos
<vorname>… </vorname>
<nachname>… </nachname>
{xlink, http://www.w3.org/1999/xlink}
xlink:href="http://www.turing.org.uk/"
xlink:type="simple"
<vorname>… </vorname>
<m…> </m…>
<nachname>… </nachname>
Alan
Turing
Richard
P
Feynman

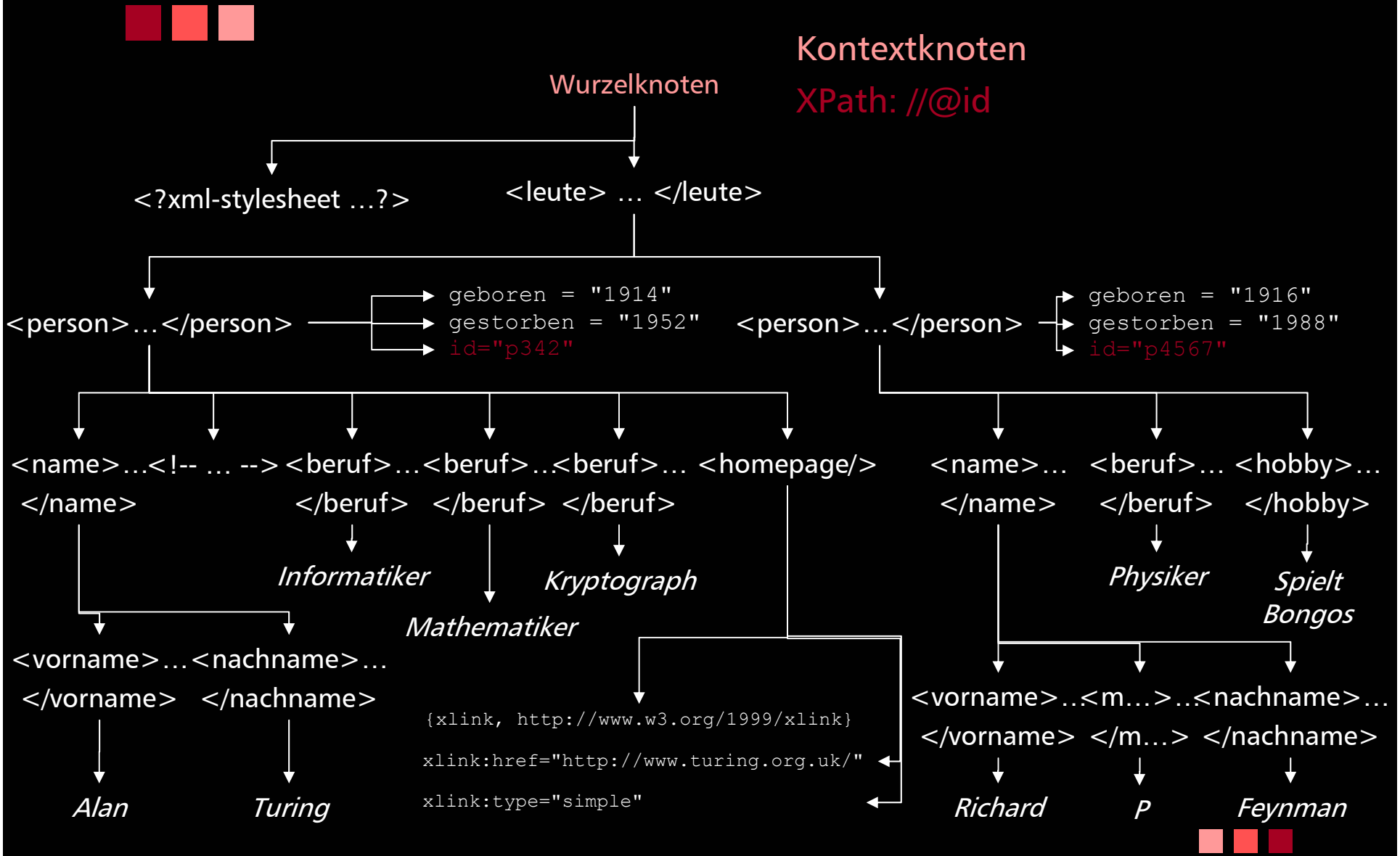
Alternativen



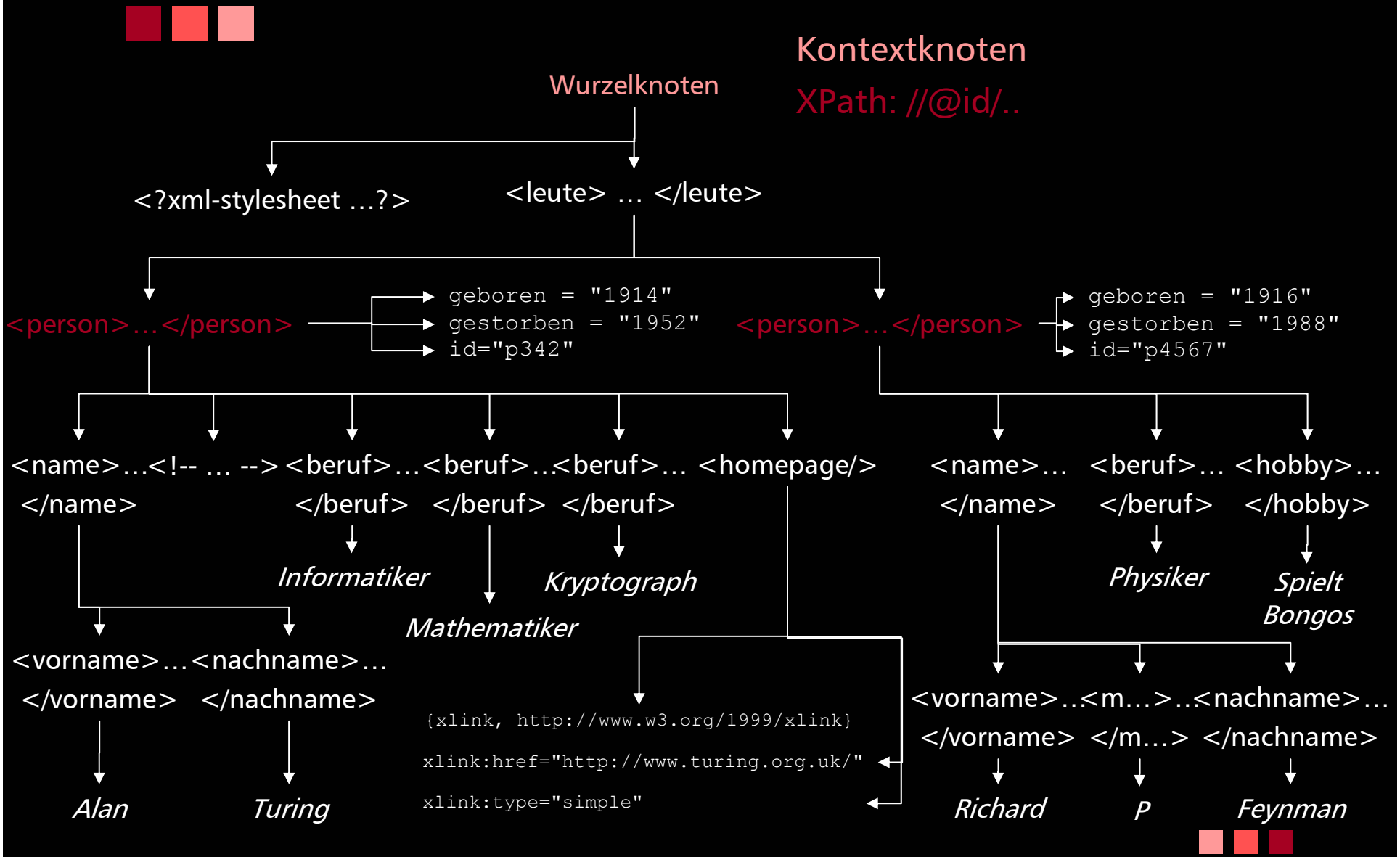
Zusammengesetzte Pfade



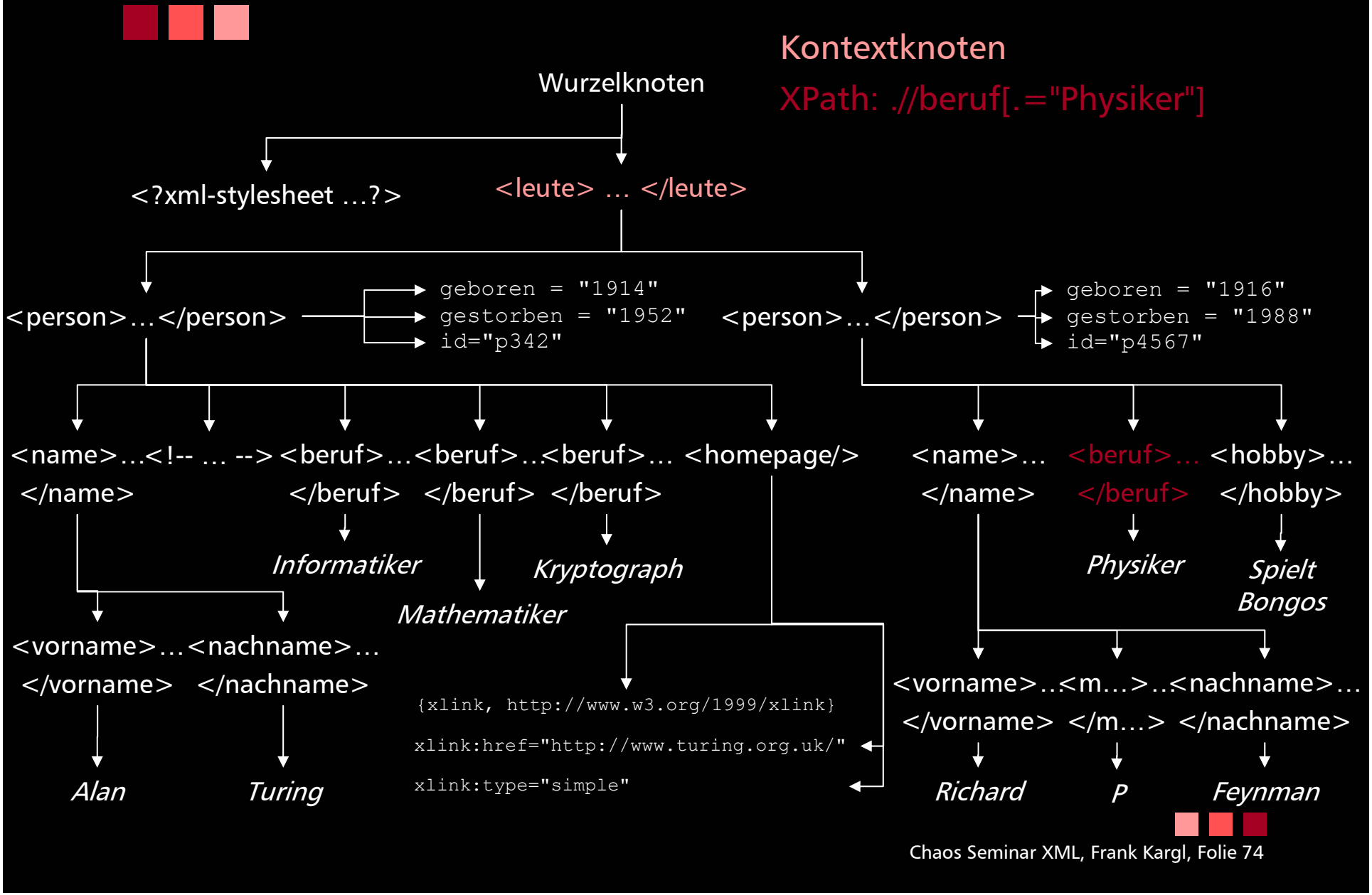
Zusammengesetzte Pfade



Zusammengesetzte Pfade



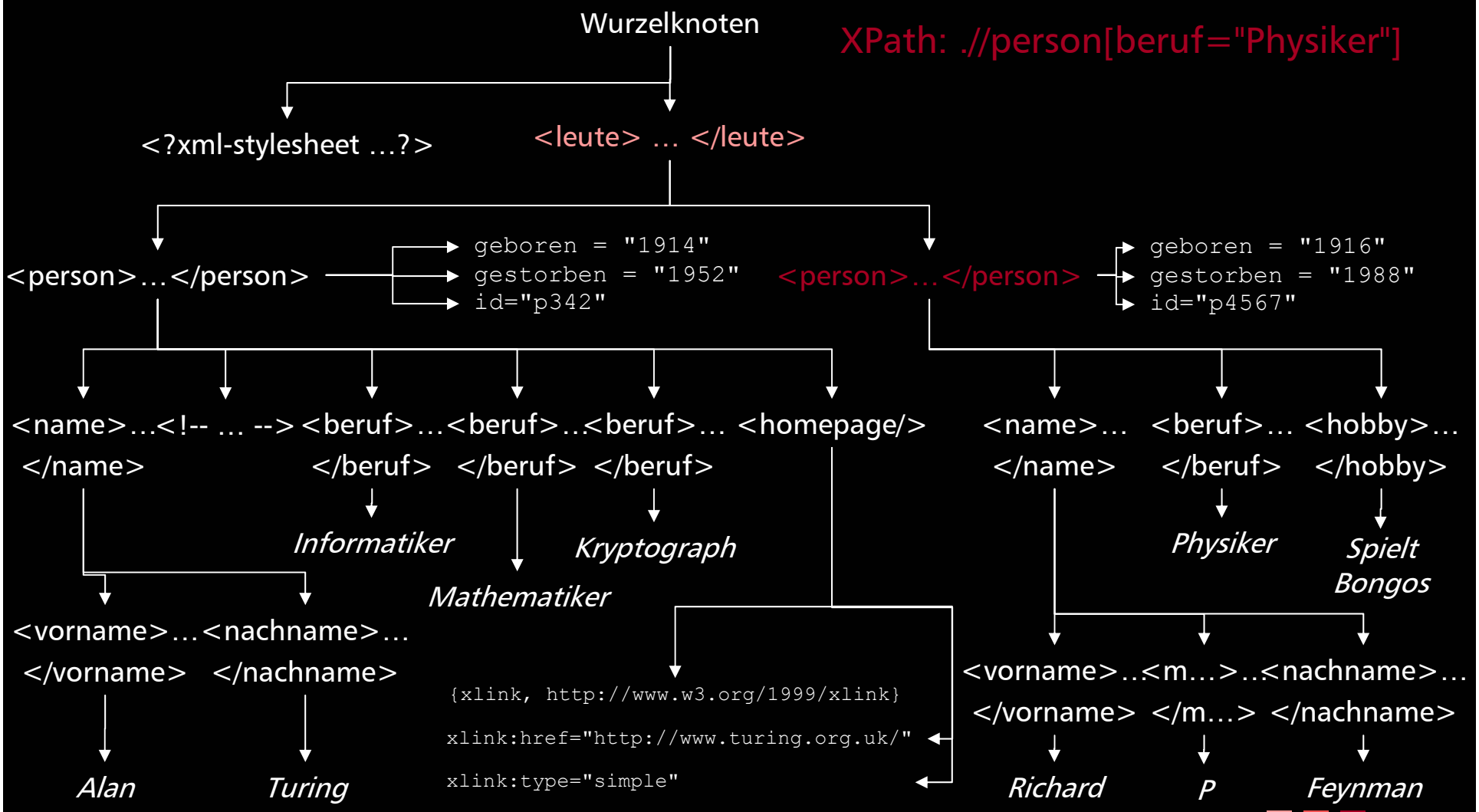
Prädikate


Kontextknoten
XPath: .//beruf[.="Physiker"]
Wurzelknoten
<?xml-stylesheet …?>
<leute> … </leute>
<person>…</person>
geboren = "1914"
gestorben = "1952"
id="p342"
<person>…</person>
geboren = "1916"
gestorben = "1988"
id="p4567"
<name>…<!-- … --> <beruf>…<beruf>…<beruf>… <homepage/>
</name> </beruf> </beruf> </beruf>
Informatiker
Kryptograph
Mathematiker
<vorname>…<nachname>…
</vorname> </nachname>
Alan
Turing
{xlink, http://www.w3.org/1999/xlink}
xlink:href="http://www.turing.org.uk/"
xlink:type="simple"
<name>… <beruf>… <hobby>…
</name> </beruf> </hobby>
Physiker
Spielt Bongos
<vorname>…<m…>…<nachname>…
</vorname> </m…> </nachname>
Richard
P
Feynman

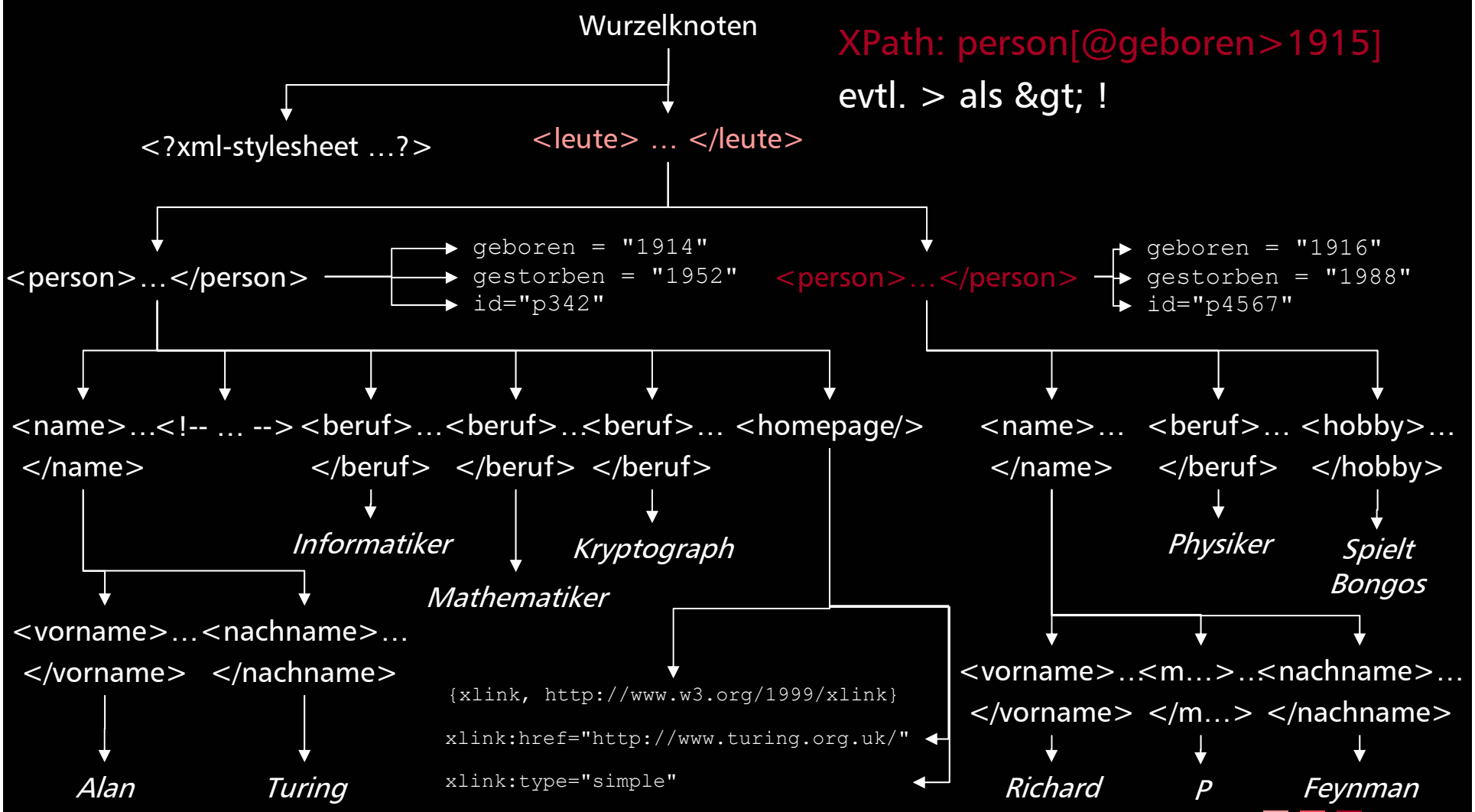
Prädikate

Kontextknoten

XPath: .//person[beruf="Physiker"]



Prädikate


Wurzelknoten
<?xml-stylesheet …?>
<leute> … </leute>
Kontextknoten
XPath: person[@geboren>1915]
evtl. > als > !
<person>…</person>
geboren = "1914"
gestorben = "1952"
id="p342"
<person>…</person>
geboren = "1916"
gestorben = "1988"
id="p4567"
<name>…<!-- … --> </name>
<beruf>… </beruf>
<beruf>… </beruf>
<beruf>… </beruf>
<homepage/>
<name>… </name>
<beruf>… </beruf>
<hobby>… </hobby>
Informatiker
Mathematiker
Kryptograph
Physiker
Spielt Bongos
<vorname>… </vorname>
<nachname>… </nachname>
{xlink, http://www.w3.org/1999/xlink}
xlink:href="http://www.turing.org.uk/"
xlink:type="simple"
<vorname>… </vorname>
<m…> </m…>
<nachname>… </nachname>
Alan
Turing
Richard
P
Feynman

XLink

- XML-Syntax zur Beschreibung gerichteter Graphen, wobei die Knoten (beliebige) Dokumente und die Kanten Links darstellen.
- Erlaubt beispielsweise
  - Unidirektionale Links
  - Bidirektionale Links
  - Multidirektionale Links
- Anwendungen
  - Externe Links
  - Inhaltsverzeichnisse
  - Register
  - uvm.

XLink

- Erweitert die Möglichkeiten von HTML <A>
- Mögliche Typen für **`xlink:type`**
  - simple, extended, locator, arc, title, resource
- Type simple: **`xlink:href`** zeigt auf URI
- Andere Typen: siehe später
- Link Verhalten
  - xlink:show=new|replace|embed|other|none
  - xlink:actuate=onLoad|onRequest|other|none
- Link-Semantic
  - xlink:title: Titel für Link
  - xlink:role: URI welche Art des Links beschreibt

Beispiel mit einfachen XLinks

```
<roman xmlns:xlink="http://www.w3.org/1999/xlink"
 xlink:type="simple"
 xlink:href="http://www.amazon.de/exec/obidos/ASIN/3453146972/"
 xlink:show="new" xlink:actuate="onRequest"
 xlink:title="Per Anhalter durch die Galaxis bei amazon.de"
 xlink:role="http://www.amazon.de/">
 <titel>Per Anhalter durch die Galaxis</titel>
 <autor>Douglas Adams</autor>
 <jahr>1978</jahr>
</roman>
<roman xmlns:xlink="http://www.w3.org/1999/xlink"
 xlink:type="simple"
 xlink:href="http://www.amazon.de/exec/obidos/ASIN/3453146980/"
 xlink:show="new" xlink:actuate="onRequest"
 xlink:title="Das Restaurant ... bei amazon.de"
 xlink:role="http://www.amazon.de/">
 <titel>Das Restaurant am Ende des Universums</titel>
 <autor>Douglas Adams</autor>
 <jahr>1980</jahr>
</roman>
<roman xmlns:xlink="http://www.w3.org/1999/xlink"
 xlink:type="simple"
 xlink:href="http://www.amazon.de/exec/obidos/ASIN/3453146050/"
 xlink:show="new" xlink:actuate="onRequest"
 xlink:title="Das Leben, das Universum ... bei amazon.de"
 xlink:role="http://www.amazon.de/">
 <titel>Das Leben, das Universum und der ganze Rest</titel>
 <autor>Douglas Adams</autor>
 <jahr>1982</jahr>
</roman>
```

Erweiterte XLinks

- **`xlink:type="extended"`**
- **`xlink:locator`**
  Element dient als Anker eines Links
- **`xlink:label`**
  Verknüpft (XML-)Namen mit Locator
- **`xlink:href`**
  Verweis auf URI, mit welchem Locator verknüpft ist
- Bögen (**`xlink:type="arc"`**) definieren Kanten zwischen zwei oder mehr Ankern

Beispiel mit erweiterten XLinks

```
<reihe xlink:type="extended" xmlns:xlink="http://www.w3.org/1999/xlink">
 <roman xlink:type="locator" xlink:label="hitchhiker1"
  xlink:href="http://www.amazon.de/exec/obidos/ASIN/3453146972/">
  <titel>Per Anhalter durch die Galaxis</titel>
  <autor>Douglas Adams</autor>
  <jahr>1978</jahr>
 </roman>
 <roman xlink:type="locator" xlink:label="hitchhiker2"
  xlink:href="http://www.amazon.de/exec/obidos/ASIN/3453146980/">
  <titel>Das Restaurant am Ende des Universums</titel>
  <autor>Douglas Adams</autor>
  <jahr>1980</jahr>
 </roman>
 <roman xlink:type="locator" xlink:label="hitchhiker3"
  xlink:href="http://www.amazon.de/exec/obidos/ASIN/3453146050/">
  <titel>Das Leben, das Universum und der ganze Rest</titel>
  <autor>Douglas Adams</autor>
  <jahr>1982</jahr>
 </roman>
 <biographie xlink:type="locator" xlink:label="dadams"
  xlink:href="http://www.douglasadams.com/dna/bio.html">
  <titel>Biographie von Douglas Adams</titel>
 </biographie>
 <vor    xlink:type="arc" xlink:from="hitchhiker1" xlink:to="hitchhiker2" />
 <vor    xlink:type="arc" xlink:from="hitchhiker2" xlink:to="hitchhiker3" />
 <zurück xlink:type="arc" xlink:from="hitchhiker2" xlink:to="hitchhiker1" />
 <zurück xlink:type="arc" xlink:from="hitchhiker3" xlink:to="hitchhiker2" />
 <info xlink:type="arc" xlink:to="dadams" xlink:title="Biographie d. Autors" />
</reihe>
```

Beispiel mit erweiterten XLinks

vor

zurück

vor

zurück

amazon.de

amazon.de

amazon.de

info

DOUGLAS ADAMS . COM

Extended XLink

XPointer

- Nicht-XML Syntax, um Positionen in XML Dokumenten anzugeben (z.B. in Links)
- In HTML:

dokument1.html

```
<a href="dokument2.html#anker">Start</a>
```

Ççâì ãÉåíOKÜí ãä

```
<a name="anker">Ziel</a>
```

- In XML:

```
http://www.kargl.net/info/
  index.xml#xpointer(//name[position=1])
```

Schemabeschreibung

- Ermöglichen die Validierung eines Dokuments
- Optionaler Schritt beim Parsing
- Alternativen:
  - Document Type Definitions (DTDs)
  - XML Schema

Document Type Definition (DTD)

- Keine XML Anwendung!
- Entspricht weitgehend SGML DTD
- Einbindung einer DTD in XML Dokument durch DOCTYPE Deklaration

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE person SYSTEM "http://www.da.de/dtd/person.dtd">
<person>
  <name>
    <vorname>Alan</vorname>
    <nachname>Turing</nachname>
  </name>
  <beruf>Informatiker</beruf>
  <beruf>Mathematiker</beruf>
  <beruf>Kryptograph</beruf>
</person>
```

Dokumenttyp-Deklaration

- Entweder privat (SYSTEM) oder öffentlich (PUBLIC)

```
<!DOCTYPE rss PUBLIC
   "-//Netscape Communications//DTD RSS 0.91//EN"
   "http://my.netscape.com/publish/formats/rss-0.91.dtd">
```

- Formal Public Identifier (FPI)
- Interne DTD Teilmengen

```
<!DOCTYPE leute SYSTEM "leute.dtd" [
  <!ELEMENT zweitname (vorname, nachname)>
]>
```

- Aber: Interne und externe DTD müssen kompatibel sein
  - Lediglich Entities dürfen umdefiniert werden

Einfaches DTD Beispiel

ÜííéWLL ï ÉÄÉåÖKáåÑçê ã~íáâK ì åáJ ì ä ã KÇÉL ï ÉÄÉåÖÇÉ ã çëLñ ã äLäÉ ì íÉKñ ã äW

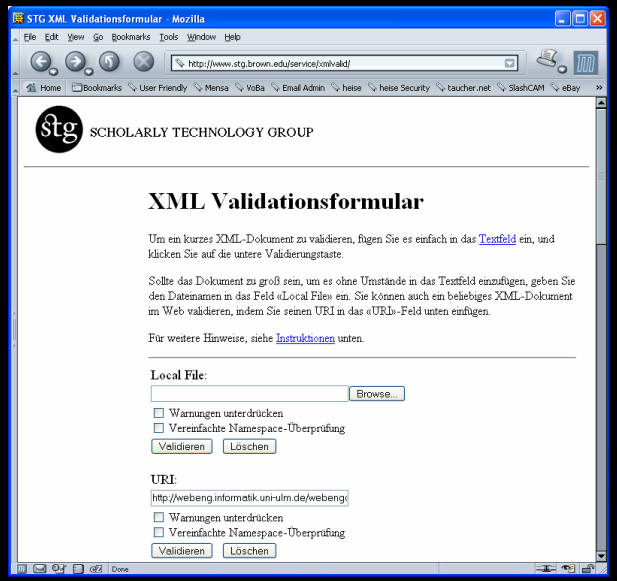
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE leute SYSTEM "leute.dtd">
<leute>
 <person geboren="1912" gestorben="1954" id="p342">
  <name>
   <vorname>Alan</vorname>
   <nachname>Turing</nachname>
  </name>
  <beruf>Informatiker</beruf>
  <beruf>Mathematiker</beruf>
  <beruf>Kryptograph</beruf>
  <homepage xlink:href="http://www.turing.org.uk/"/>
 </person>
 <person geboren="1918" gestorben="1988" id="p4567">
  <name>
   <vorname>Richard</vorname>
   <mittelinitial>P</mittelinitial>
   <nachname>Feynman</nachname>
  </name>
  <beruf>Physiker</beruf>
  <hobby>Spielt Bongos</hobby>
 </person>
</leute>
```

Einfaches DTD Beispiel

Ü í í é W L L ï É Ä É å Ö K á å Ñ ç ê ã ~ í á â K ì å á J ì ä ã K Ç É L ï É Ä É å Ö Ç É ã ç ë L ñ ã ä L ä É ì í É K Ç í Ç W

```
<!ELEMENT leute     (person+)>
<!ELEMENT person    (name, (beruf|hobbies)*, homepage?)>
<!ATTLIST person    geboren CDATA #REQUIRED
                    gestorben CDATA #REQUIRED
                    id ID #IMPLIED>
<!ELEMENT name      (vorname, mittelinitial?, nachname)>
<!ELEMENT vorname   (#PCDATA)>
<!ELEMENT mittelinitial   (#PCDATA)>
<!ELEMENT nachname  (#PCDATA)>
<!ELEMENT beruf     (#PCDATA)>
<!ELEMENT hobbies   (#PCDATA)>
<!ELEMENT homepage  EMPTY>
<!ATTLIST homepage  xmlns:xlink CDATA #FIXED "http://www.w3.org/1999/xlink"
                    xlink:type CDATA #FIXED "simple"
                    xlink:href CDATA #REQUIRED>
```

Validieren von Dokumenten

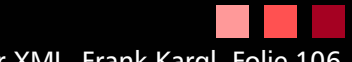


- Online Validierer
  - z.B. http://www.stg.brown.edu/service/xmlvalid/
- Mittels Xerces XML Parser
- Separate Applikation
  - z.B. Schematron Validator von Topologi (http://www.topologi.com/products/validator/index.html)
- Validierender XML Parser
  - z.B. sax.Counter aus Xerces Parser

```
java -classpath xercesSamples.jar sax.Counter -v leute.xml
leute.xml: 318 ms (16 elems, 8 attrs, 71 spaces, 81 chars)
```

Nachteile von DTDs

- Harmonieren schlecht mit Namensräumen
- Keine XML Anwendung,
  eigener Parser notwendig
- Nur eine DTD pro Dokument
- Vielzahl von Einschränkungen
  - Anzahl von Elementen schwer festlegbar
    (z.B. genau drei A und drei B Elemente in
    beliebiger Reihenfolge)
  - Genaue Typen von Inhalten nicht festlegbar
  - uvm.
- Mächtigere Schemasprache notwendig



XML Schema

- Verabschiedet 2.5.2002
- Begriffe
  - *(W3C-)XML-Schema*: Beschreibung einer XML Anwendung
  - *Instanz-Dokument*: XML Dokument, welches bzgl. eines XML-Schemas gültig ist
  - *Gültig im Sinne des (XML-)Schemas*: Ein Dokument ist ein Instanz-Dokument eines XML-Schemas

XML-Schema vs. DTD

DTDs:

- Verschachtelung von Elementen
- Häufigkeitsbeschränkung von Elementen
- Zulässige Attribute
- (einfache) Attributtypen und Default-Werte
- Definition von Entities

XML-Schema:

- Einfache und komplexe Datentypen
- Ableitung und Vererbung von Typen
- Häufigkeitsbeschränkung von Elementen
- Namensraumsensitive Element- und Attributdeklarationen
- KEINE allgemeinen Entities!

Einfaches XML-Schema Beispiel

- XML Dokument:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vollständigerName
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="address-schema.xsd">
Frank Kargl
</vollständigerName>
```

- XML-Schema Dokument:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="vollständigerName" type="xs:string"/>
</xs:schema>
```

Validierung

- Xerces

```
$ java -classpath xercesSamples.jar dom.Writer -v -n -s -f
   address-schema.xml
<vollständigerName
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="address-schema.xsd">
  Frank Kargl
</vollständigerName>
```

```
$ java -classpath xercesSamples.jar dom.Writer -v -n -s -f
   address-schema-wrong.xml
[Error] address-schema-wrong.xml:6:21: cvc-type.3.1.2:
   Element 'vollständigerName' must have no element
   information item [children].
<?xml version="1.0" encoding="UTF-8"?>
<vollständigerName
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="address-schema.xsd">
Frank <b>Kargl</b>
</vollständigerName>
```

Komplexes XML-Schema Beispiel

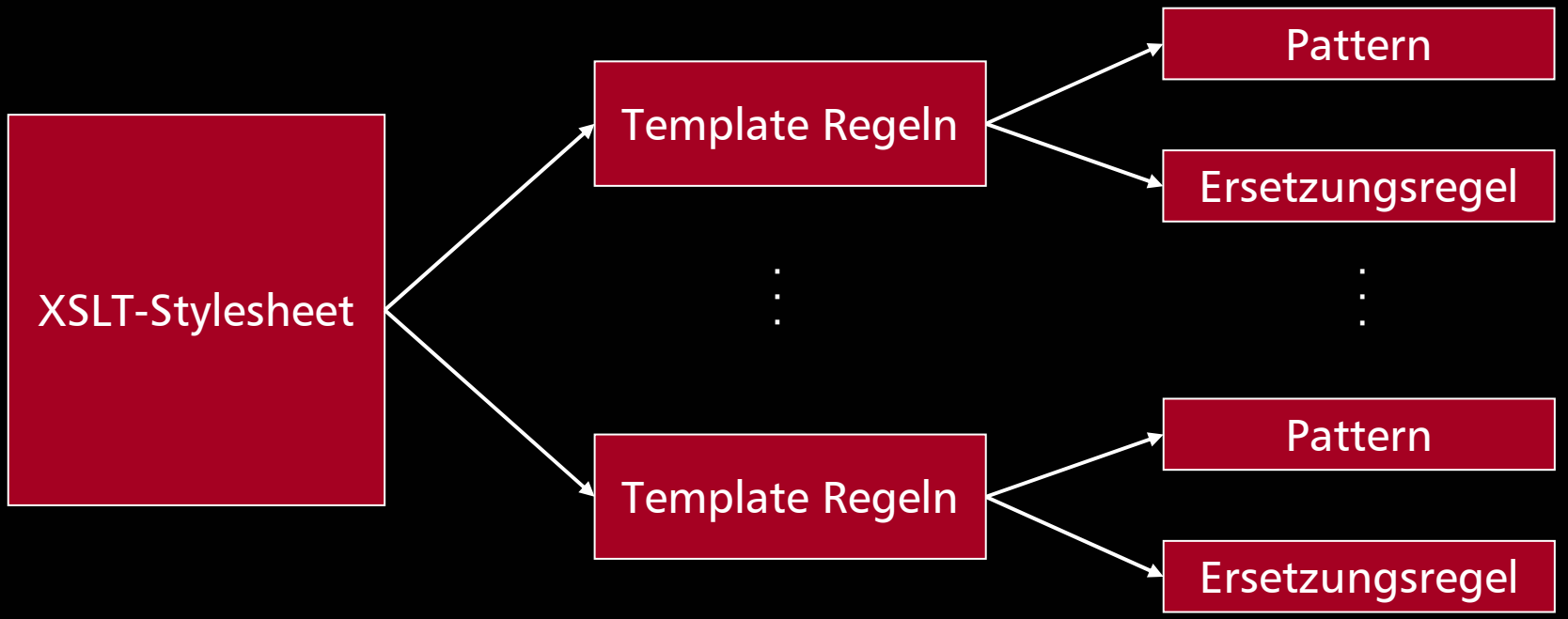
- Wollt ihr das wirklich sehen???

Transformationen

- Möglichkeiten zur Transformation von XML Dokumenten in ein Ausgabeformat
  - Explizit programmieren
  - Cascading StyleSheets (direkte Formatierung)
  - Extensible Stylesheet Language
    - XSLT (Transformations)
    - XSL-FO (Formatting Objects)
- Unterscheidung
  - Transformation: XML → XML
  - Formatierung: XML → * (z.B. PDF, TeX, Bitmap)

XSLT

- Eine XML Anwendung, welche die Transformation von einem XML Dokument in ein anderes beschreibt.



Minimales XSLT Stylesheet

```
<?xml version="1.0"?>
<leute>
 <person geboren="1912" gestorben="1954" id="p342">
  <name>
   <vorname>Alan</vorname>
   <nachname>Turing</nachname>
  </name>
  <beruf>Informatiker</beruf>
  <beruf>Mathematiker</beruf>
  <beruf>Kryptograph</beruf>
 </person>
 <person geboren="1918" gestorben="1988" id="p4567">
  <name>
   <vorname>Richard</vorname>
   <mittelinitial>P</mittelinitial>
   <nachname>Feynman</nachname>
  </name>
  <beruf>Physiker</beruf>
  <hobby>Spielt Bongos</hobby>
 </person>
</leute>
```

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
</xsl:stylesheet>
```

Xalan Stylesheet Processor

```
$ java org.apache.xalan.xslt.Process -IN leute.xml -XSL minimal.xslt
<?xml version="1.0" encoding="UTF-8"?>


        Alan
        Turing

      Informatiker
      Mathematiker
      Kryptograph



        Richard
        P
        Feynman

      Physiker
      Spielt Bongos
```

XSLT Stylesheets

- Angabe des Stylesheets im XML Dokument

```
<?xml version="1.0"?>
<?xml-stylesheet type="application/xml" href="minimal.xsl"?>
<leute>
 ...
```

- Template Regeln
  - Allgemeine Form
  - **<xsl:template match="***XPath-Muster***">** *Ersetzung-Template* **</xsl:template>**

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="person">
  <p>Eine Person</p></xsl:template>
</xsl:stylesheet>
```

  - Ersetzungs-Template muss wohlgeformt sein!

XSLT Stylesheets

- Übernahme von Werten aus Ursprungsdokument

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="person">
    <p>Person:
    <xsl:value-of select="name/nachname"/></p>
  </xsl:template>
</xsl:stylesheet>
```

- XPath Ausdruck zum Selektieren der Werte

XSLT Stylesheets

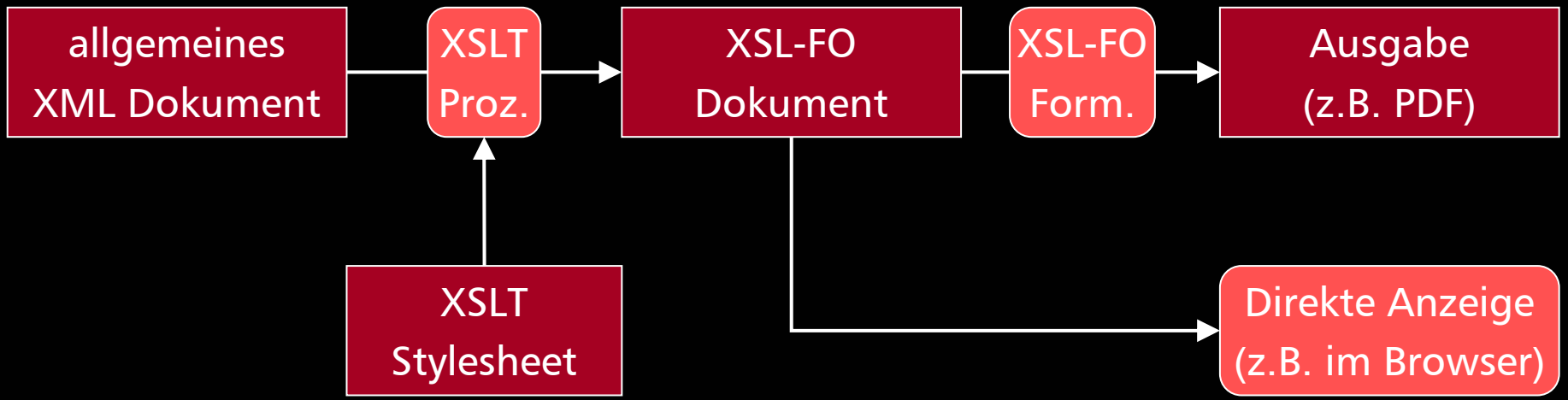
- Verarbeitungsreihenfolge festlegen

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="leute">
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head><title>Berühmte Wissenschaftler</title></head>
      <body>
        <xsl:apply-templates/>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="name">
    <p><xsl:value-of select="nachname"/>, <xsl:value-of select="vorname"/></p>
  </xsl:template>
  <xsl:template match="person">
    <xsl:apply-templates select="name"/>
  </xsl:template>
</xsl:stylesheet>
```

- Problem: DOCTYPE Deklaration im Ersetzungs-Template nicht erlaubt!

XSL-Formatting Objects (XSL-FO)

- Dient der genauen Beschreibung eines Seitenlayouts mittels XML
- Übliche Verarbeitungskette



XSL-FO

- Formatierungsprogramm
  - Beispiel FOP von Apache Group (http://xml.apache.org/fop/)
- Elemente
  - Block-Bereiche
  - Inline-Bereiche
  - Zeilen-Bereiche
  - Symbol-Bereiche

XSL-FO Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
 <fo:layout-master-set>
  <fo:simple-page-master margin-right="1in" margin-left="1in" margin-bottom="1in"
   margin-top="1in" page-width="8.5in" page-height="11in" master-name="first">
        <fo:region-body/>
  </fo:simple-page-master>
 </fo:layout-master-set>
 <fo:page-sequence master-name="first">
  <fo:flow flow-name="xsl-region-body">
   <fo:block font-family="Times, serif" font-size="12pt">
    <fo:block font-family="Helvetica, Arial, sans-serif" font-size="20pt"
     font-weight="bold" text-align="center">Southern Corn Bread</fo:block>
    <fo:list-block>
     <fo:list-item><fo:list-item-label><fo:block>•</fo:block></fo:list-item-label>
      <fo:list-item-body><fo:block>
       1 cup flour
      </fo:block></fo:list-item-body></fo:list-item>
     [... Weitere Zutaten ...]
    </fo:list-block>
    <fo:block margin-top="12pt" margin-left="4pt">
     Sift flour, baking powder, sugar &amp;amp; salt together.
     Add 1 cup corn meal. [... usw. ...]
    </fo:block>
   </fo:block>
  </fo:flow>
 </fo:page-sequence>
</fo:root>
```

Vergleich XSL-FO vs. CSS

- Vorteile CSS
  - Einfache Syntax
  - Etabliert
  - Ideal insbes. für Präsentation im Web
- Vorteile XSL-FO
  - XSL-FO besser für Druck geeignet
  - Vielfältige Möglichkeiten
    - Mehrspaltige Layouts
    - Fußnoten
    - Seitenzahlen
    - Querverweise

Programmiermodelle für XML

- Textorientiert
  - Direktes Lesen/Manipulieren der Textdatei
  - Reguläre Ausdrücke, Suchen/Ersetzen etc.
  - Allgemeine Tools: z.B. vi, sed, awk, perl, …
  - Standardwerte für Attribute? DTDs? Entities? Namensräume?
- Bewertung
  - Editieren und Lesen von Dokumenten mittels Texteditoren etc. häufig
  - Verarbeiten von XML Dokumenten mit allgemeinen Tools schwierig und fehleranfällig

Programmiermodelle für XML

- Ereignisorientiert
  - XML-Parser liest Dokument sequentiell
  - Beim Auftreten von bestimmten Bedingungen (z.B. Start Tag) werden Ereignisse generiert und via Callbacks an Applikation gesendet
  - Kontext (Namensräume, Gültigkeit usw.) wird mitgeführt
  - Immer nur ein Teil des Dokuments (+ Kontext) im Speicher
- Bewertung
  - Schnell und effizient
  - Kann auch sehr große Dokumente verarbeiten
  - Nie ganzes Dokument auf einmal verfügbar (z.B. zur Analyse von XPath Ausdrücken in Attributen)
  - Oft: Anwendung selbst speichert Zustand

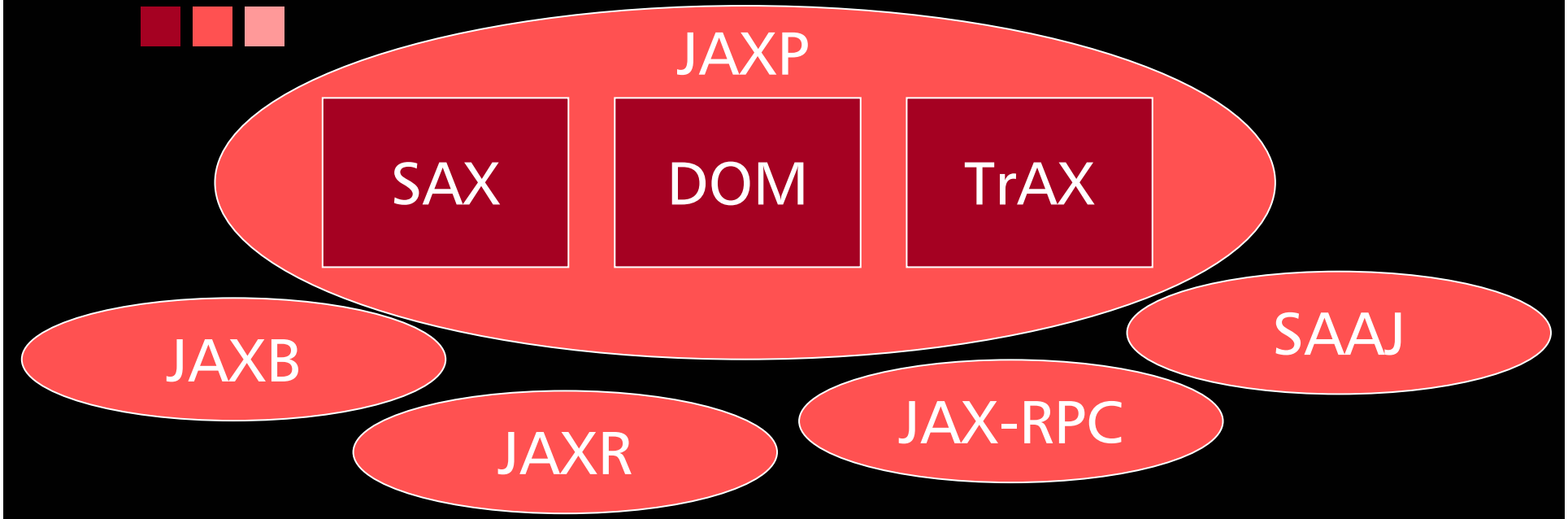
Programmiermodelle für XML

- Baum- oder objektorientiert
  - XML-Parser liest komplett ein
  - Interne Darstellung als Baum- oder Objektstruktur
  - Zugriff durch Navigationsmethoden
- Bewertung
  - Verbraucht evtl. viel Speicher
  - Erlaubt freie Navigation im Baum
  - Unterstützung von XPath u.U. bereits integriert

Programmiermodelle für XML

- Transformation
  - Z.B. mittels XSLT
  - Keine Programmierung im engeren Sinne
- Implizit
  - Z.B. bei XML-RPC
  - Kein direkter Kontakt des Programmierers mit XML
- Bewertung
  - Für Spezialaufgaben durchaus geeignet
  - Keine Einarbeitung in XML notwendig(?)

Übersicht Java APIs



- JAXP: Java API for XML Processing
  - SAX: Simple API for XML
  - DOM: Document Object Model
  - TrAX: Transformation API for XML
- JAXB: Java API for XML Binding
- JAXR: Java API for XML Registries
- JAX-RPC: Java API for XML Remote Proc. Call
- SAAJ: SOAP with Attachments API for Java

Document Object Model

- Programmiersprachen-unabhängiges Modell zum Zugriff auf Markup-Dokumente
- Allgemeiner und (HTML- bzw. XML-) spezifischer Teil
- Beschreibung der Schnittstellen in Corba-IDL
- Language-Bindung für Java, C, C++, Scheme uvm.
- DOM XML Baum ähnlich wie bei XPath

DOM Grundlagen

- Initialisieren und Parsen eines Dokuments

```
DocumentBuilderFactory factory =
   DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
document = builder.parse(new File(args[0]));
```

- Parametrisieren

```
factory.setValidating(true);
factory.setIgnoringElementContentWhitespace(true);
factory.setNamespaceAware(true);
factory.setIgnoringComments(true);
```

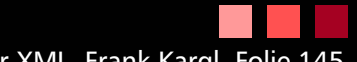
DOM Grundlagen

- Zugriff auf Knoten

```
static void printTree(Node node, int level) {
        for (int i = 0;i < level;i++) {System.out.print('-');}
        System.out.print(node.getNodeName());
        if (node.getNodeValue() != null) {
            System.out.print(": " + node.getNodeValue());
        }
        if (node.hasAttributes()) {
            System.out.print(" # ");
            NamedNodeMap attributes = node.getAttributes();
            for (int j = 0;j < attributes.getLength();j++) {
                Node attribute = attributes.item(j);
                System.out.print(attribute.getNodeName()+"="+
                          attribute.getNodeValue()+";");
            }
        }
        System.out.println();
        NodeList nl = node.getChildNodes();
        for (int k = 0;k < nl.getLength();k++) {
            Node subNode = nl.item(k);
            printTree(subNode, level + 1);
        }
    }
```

DOM Entwicklung

- DOM Level 1
  - Grundsätzliche Verarbeitung von HTML/XML Dokumenten
- DOM Level 2
  - Modularisierung
  - Views, Events, Style, Traversal, Range
- DOM Level 3
  - Ein-/Ausgabe von Dokumenten
  - Integration von Validierung
  - DTDs und XML Schema
  - XPath



Simple API for XML

- Industriestandard für XML Parsing in Java
- Ereignisorientiert
- Mittlerweile auch Umsetzungen für andere Programmiersprachen (C++, Python, Perl, Eiffel, ...)
- Wichtig für DOM Programmierer:
  DOM Fehlerbehandung benutzt
  SAX Exceptions

SAX Grundlagen

- Einlesen eines Dokuments

```
SAXParserFactory spf = SAXParserFactory.newInstance();
SAXParser sp = spf.newSAXParser();
XMLReader reader = sp.getXMLReader();

InputSource inputSource = new InputSource(uri);
reader.parse(inputSource);
```

- Setzen der Callback-Handler

```
public class SimpleSAX implements ContentHandler,
   ErrorHandler, LexicalHandler {
    ...
    reader.setContentHandler(this);
    reader.setErrorHandler(this);
    ...
}
```

SAX Features und Properties

- Steuern das Verhalten des Parsers
  - Features: Binär
  - Properties: Wert

```
String featureURI = "http://xml.org/sax/features/validation";
reader.setFeature(featureURI, true);
boolean state = true;
String featureURI = "http://xml.org/sax/features/namespaces";
reader.setFeature(featureURI, state);
featureURI = "http://xml.org/sax/features/namespace-prefixes";
reader.setFeature(featureURI, !state);
reader.setProperty("http://xml.org/sax/properties/lexical-handler",
   this);
```

SAX Callback Handler

- ContentHandler
- DefaultHandler liefert leere Methoden (für alle Callback Handler)

```
public void setDocumentLocator(Locator locator) {}
public void startDocument() throws SAXException {}
public void endDocument() throws SAXException {}
public void startPrefixMapping(String prefix, String uri) {}
public void endPrefixMapping(String prefix) {}
public void startElement(String namespaceURI, String localName,
   String qName, Attributes atts) {}
public void endElement(String namespaceURI, String localName,
   String qName) {}
public void characters(char ch[], int start, int length) {}
public void ignorableWhitespace(char ch[], int start, int length) {}
public void processingInstruction(String target, String data) {}
public void skippedEntity(String name) {}
```

SAX Callback Handler

- ErrorHandler

```
public void warning(SAXParseException exception) {}
public void error(SAXParseException exception) {}
public void fatalError(SAXParseException exception){}
```

- LexicalHandler

```
public void startDTD(String name, String publicId, String systemId) {}
public void endDTD() {}
public void startEntity(String name) {}
public void endEntity(String name) {}
public void startCDATA() {}
public void endCDATA() {}
public void comment(char ch[], int start, int length) {}
```

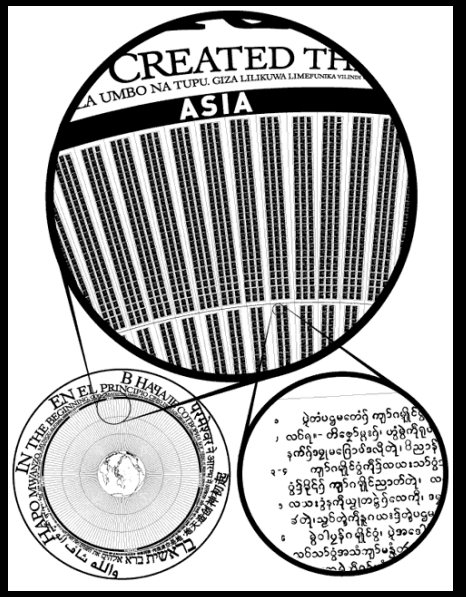
- Weitere Features: Filter, Transformer …

Bewertung XML

- "Einfache", erweiterbare Markupsprache
- Vorteile
  - Standardisierung
  - Verfügbarkeit effizienter Parser + Tools
  - Lesbarkeit
  - Unicode Unterstützung
- Nachteile
  - Größe
  - Kodier-/Dekodieraufwand

Problem Größe

- Binary XML
  - Binäre Tokens für Tags
  - Problem: Lesbarkeit
    [http://www.xml.com/pub/a/2003/08/13/deviant.html]
- Kompression
  - gzip solves it all




```
[fkargl@nathan fkargl]$ ll -h demo.xml*
-rw-------    1 fkargl   users       1.6M May 10 16:01 demo.xml
-rw-------    1 fkargl   users       142K May 10 16:02 demo.xml.bz2
-rw-------    1 fkargl   users       204K Jan 21 10:25 demo.xml.gz
```

Problem Lesbarkeit

- Das Format ist lesbar
- Der Datenträger auch???
- XML lässt sich ausdrucken und abheften!

The End

Noch Fragen?

http://www.xml.com/