



# Aufbau von XML-Dokumenten



## Heutige Vorlesung

- Syntax wohlgeformter XML-Dokumente
- Namensräume zur Auflösung von Namenskonflikten
- Festlegung der Semantik von XML-Elementen

## Wiederholung: Was ist XML?

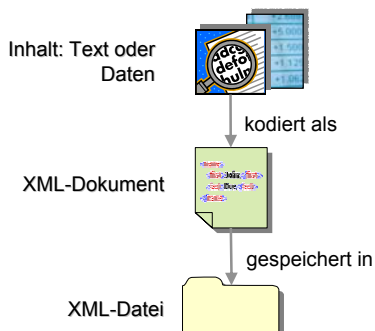


- XML ist eine Methode, um strukturierte Daten in einer Textdatei darzustellen.
- XML sieht fast aus wie HTML, ist aber kein HTML.
- XML ist Text, aber nicht zum Lesen.
- XML ist eine Familie von Techniken.
- XML ist neu, aber nicht so neu.
- XML ist lizenzfrei, plattformunabhängig und gut unterstützt.



# Syntax wohlgeformter XML-Dokumente

## Was ist ein XML-Dokument?



## Grundbausteine von XML



- Elemente: strukturieren das XML-Dokument
- Attribute: Zusatzinformationen zu einzelnen Elementen
- Die XML-Deklaration: Informationen für Parser

```
<?xml version="1.0" encoding="UTF-8"?>
<name id="1232345">
  <first>John</first>
  <middle>Fitzgerald Johansen</middle>
  <last>Doe</last>
</name>
```

- Namensräume: lösen Namenskonflikte auf und geben Elemente eine bestimmte Bedeutung

## Grundbausteine von XML: Elemente

- haben einen Namen
- bestehen aus:
  - einem Anfangs-Tag (engl. *start tag*)
  - einem dazugehörigen Ende-Tag (engl. *end tag*)
  - einem Inhalt.
- **Beispiel:** `<first>John</first>`
- „first“ ist der *Name* des Elementes.
- `<first>` ist ein *Anfangs-Tag*.
- `</first>` ist das dazugehörige *Ende-Tag*.
- „John“ ist der *Inhalt* des Elementes.
- `<first>John</first>` wird *Element* genannt.

© Klaus Schild, 2004

7

## Inhalt von Elementen

vier verschiedene Arten von Inhalt:

- **unstrukturierter Inhalt:**  
einfacher Text (String)
- **strukturierter Inhalt:**  
enthält weitere Elemente
- **gemischter Inhalt:**  
enthält gleichzeitig Elemente und Text
- **leerer Inhalt**

© Klaus Schild, 2004

8

## Unstrukturierter Inhalt

- **Beispiel:** `<first>John</first>`
- wird auch als *parsed character data* (PCDATA) bezeichnet:
  - *character data*: es handelt sich um einfachen Text
  - *parsed*: Text wird vom Parser analysiert, um das Ende-Tag zu identifizieren.
- **Beachte:** Für XML reservierte Symbole wie `<` und `&` dürfen in PCDATA *nicht* verwendet werden.
- Wie in HTML stattdessen `&lt;` bzw. `&amp;` verwenden.
- `&lt;` und `&amp;` werden in XML auch *entity references* genannt.

© Klaus Schild, 2004

9

## Unstrukturierter Inhalt: CDATA

- Unstrukturierter Inhalt mit vielen reservierten Symbolen besser als sog. *character data* (CDATA) darstellen.
- **Beispiel:**  
`<formula>`  
`<![CDATA[ X > Y & Y > Z ]]>`  
`</formula>`
- Inhalt: String zwischen inneren Klammern  
hier: `X > Y & Y > Z`
- XML-Parser sucht in CDATA lediglich die Zeichenkette `]]>`, analysiert den Inhalt aber ansonsten nicht.

© Klaus Schild, 2004

10

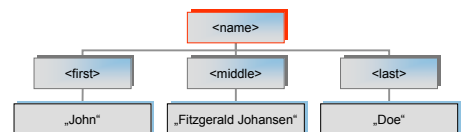
## Strukturierter Inhalt

- **Beispiel:**  
`<name>`  
`<first>John</first>`  
`<last>Doc</last>`  
`</name>`
- **Kind-Elemente:** Elemente, die im Inhalt eines Elementes vorkommen  
hier: `<first>John</first>` und `<last>Doc</last>`
- Elemente können beliebig tief geschachtelt werden.

© Klaus Schild, 2004

11

## Baumstruktur von XML



- Jedes XML-Dokument hat *genau ein* Wurzelement.
- Kind-Elemente sind immer *geordnet*.

© Klaus Schild, 2004

12

## Gemischter Inhalt

- enthält *gleichzeitig* mindestens ein Kind-Element und Text (PCDATA oder CDATA)

- Beispiel:**

```
<section>
  Text
  <subsection> ... </subsection>
  Text
</section>
```

- engl. *mixed content*

## Leerer Inhalt

- Beispiel:**

```
<name>
  <first>John</first>
  <middle></middle>
  <last>Doe</last>
</name>
```

- wird auch als *leeres Element* bezeichnet
- Abkürzung:** einfach `<middle/>` statt `<middle></middle>`
- `<middle/>` wird auch selbstschließendes Element genannt

## Warum leere Elemente?

```
<name>
  <first>John</first>
  <last>Doe</last>
</name>
```

VS.

```
<name>
  <first>John</first>
  <middle></middle>
  <last>Doe</last>
</name>
```

- Kind-Element `middle` fehlt
- evtl. inkompatibel zu einer DTD oder einem XML-Schema
- Kind-Element `middle` vorhanden
- wird evtl. von einer DTD oder einem XML-Schema vorgeschrieben

## Grundbausteine von XML: Attribute

```
<name id="1232345" nickname="Shiny John">
  <first>John</first>
  <middle>Fitzgerald Johansen</middle>
  <last>Doe</last>
</name>
```

- Element kann eine beliebige Anzahl von Attributen haben.
- Attribute:** Name-Wert-Paare der Form `name="wert"` oder `name='wert'`.
- Wert eines Attributes *immer* vom Typ String (PCDATA)
- keine für XML reservierten Symbole erlaubt
- Beachte:** Reihenfolge der Attribute belanglos

## Element statt Attribut

- Jedes Attribut kann auch als Kind-Element repräsentiert werden:

```
<name id="0678038">
  <first>John</first>
  <middle>Fitzgerald</middle>
  <last>Doe</last>
</name>
```

id als Attribut



```
<names>
  <id id="0678038"></id>
  <first>John</first>
  <middle>Fitzgerald</middle>
  <last>Doe</last>
</names>
```

id als Kind-Element

## Attribut statt Element

- Jedes Kind-Element mit *unstrukturiertem* Inhalt kann auch als Attribut dargestellt werden:

```
<name>
  <id id="0678038"></id>
  <first>John</first>
  <middle>Fitzgerald</middle>
  <last>Doe</last>
</name>
```

id, first, middle und last als Kind-Elemente



```
<name id="0678038"
  first="John"
  middle="Fitzgerald"
  last="Doe" />
```

id, first, middle und last als Attribute

**Resultat:** leeres Element

## Attribut oder Element?

- Attribut kann nur einen String als Wert haben, ein Element kann beliebig strukturiert werden.
- Reihenfolge der Attribute belanglos, diejenige von Elementen nicht
- Einheitliche Darstellung mit Elementen eleganter, Darstellung mit Attributen kompakter

Fazit: Attribute eignen sich besonders für einfache, unstrukturierte Zusatzinformationen (Metadaten).

## Beispiel

```
<name id="0678038" creation-date="21.05.2003">
  <first>John</first>
  <middle>Fitzgerald</middle>
  <last>Doe</last>
</name>
```

interner Schlüssel  
des Datensatzes

Erstellungsdatum  
des Datensatzes

- Schlüssel und Erstellungsdatum sind Zusatzinformationen (Metadaten).
- Reihenfolge ist egal
- deshalb Repräsentation als Attribute
- Problem: Datum "21.05.2003" ist unstrukturierter String.

## Grundbausteine von XML: Deklaration

```
<?xml version="1.0" encoding="UTF-8"?>
<name id="1232345">
  <first>John</first>
  <middle>Fitzgerald Johansen</middle>
  <last>Doe</last>
</name>
```

- enthält Informationen für Parser, insbesondere die verwendete XML-Version und Kodierung
- muss am Anfang der Datei stehen
- ist *optional*, sollte aber dennoch immer vorhanden sein!

## Die XML-Deklaration

### version

- verwendete XML-Version
- aktuelle Version: "1.0"
- obligatorisch

### standalone

- Gibt an, ob es eine zugehörige DTD oder ein XML-Schema gibt ("no") oder nicht ("yes").
- optional

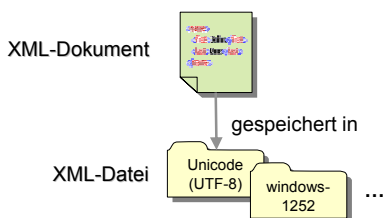
### encoding

- Kodierung der XML-Datei
- optional

Beachte: diese Attribute immer in dieser Reihenfolge

```
<?xml version="1.0" encoding="UTF-8"?>
<name id="1232345">
  <first>John</first>
  <middle>Fitzgerald Johansen</middle>
  <last>Doe</last>
</name>
```

## XML-Deklaration: Kodierung



## XML-Deklaration: Kodierung

- Intern müssen alle XML-Parser mit Unicode (UTF-8) arbeiten.
- Unicode kann *alle* nationalen Zeichen darstellen.
- encoding-Attribut gibt an, welches Kodierungsschema die betreffende XML-Datei verwendet.
- Fehlt das Attribut, dann wird angenommen, dass die XML-Datei in Unicode kodiert ist.
- Tipp: Das XML-Dokument in Unicode abspeichern. Das Attribut encoding kann dann weggelassen werden.

## Regeln für wohlgeformte XML-Dokumente

1. Jedes Anfangs-Tag muss ein zugehöriges Ende-Tag haben.
2. Elemente dürfen sich nicht überlappen.
3. XML-Dokumente haben genau ein Wurzel-Element.
4. Element-Namen müssen bestimmten Namenskonventionen entsprechen.
5. XML beachtet grundsätzlich Groß- und Kleinschreibung.
6. XML belässt Formatierungen (*white spaces*) im Text.
7. Ein Element darf niemals zwei Attribute mit dem selben Namen haben.

© Klaus Schild, 2004

25

## Regel 1: Anfangs- und Ende-Tags

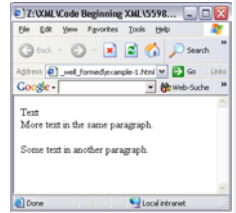
- **Jedes Anfangs-Tag muss ein zugehöriges Ende-Tag haben.**

- In HTML gilt diese Regel nicht:

```
<HTML>
<BODY>
  <P>Text
  <BR>More text in the same paragraph.
  <P>Some text in another paragraph.</P>
</BODY>
</HTML>
```

- Wo endet das erste P-Element?

→ HTML kann mehrdeutig sein.



© Klaus Schild, 2004

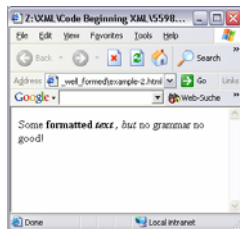
26

## Regel 2: Überlappung von Elementen

- **Elemente dürfen sich nicht überlappen.**

- In HTML gilt diese Regel nicht:

```
<HTML>
<BODY>
  <P>Some
  <STRONG>formatted
  <EM>text
  </STRONG>, but
  </EM>
  no grammar no good!
</P>
</BODY>
</HTML>
```



→ HTML kann unstrukturiert sein.

© Klaus Schild, 2004

27

## Regel 3: Wurzel-Elemente

- **Jedes XML-Dokumente hat genau ein Wurzel-Element.**

- Also z.B. statt zweier Wurzel-Elemente

```
<name>John</name>
<name>Jane</name>
```

ein zusätzliches Eltern-Element einführen:

```
<names>
  <name>John</name>
  <name>Jane</name>
</names>
```

oder

```
<employees>
  <name>John</name>
  <name>Jane</name>
</employees>
```

© Klaus Schild, 2004

28

## Regel 4: Namenskonventionen

Element- und Attribut-Namen:

- beginnen entweder mit einem Buchstaben oder „\_“:  
z.B. first, First oder \_First
- Nach dem ersten Zeichen zusätzlich Zahlen sowie „-“ und „.“ erlaubt:  
z.B. \_1st-name oder \_1st.name
- enthalten keine Leerzeichen
- enthalten kein „.“
- beginnen nicht mit „xml“, unabhängig davon, ob die einzelnen Buchstaben groß- oder kleingeschrieben sind

© Klaus Schild, 2004

29

## Beispiele

- <résumé> ✓
- <xml-tag> kein korrekter Name: beginnt mit „xml“
- <123> kein korrekter Name: beginnt mit Zahl
- <fun=xml> kein korrekter Name: enthält „=“ (erlaubt wären: \_ , - und .)
- <first name> kein korrekter Name: enthält Leerzeichen

© Klaus Schild, 2004

30

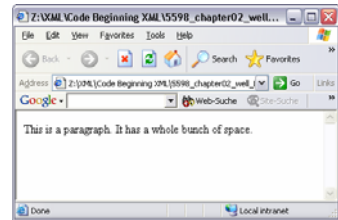
## Regel 5: Groß- und Kleinschreibung

- **XML beachtet grundsätzlich Groß- und Kleinschreibung.**
- Im Gegensatz zu HTML unterscheidet XML also z.B. zwischen `<P>` und `<p>`.

Dennoch möglichst nicht gleichzeitig  
`<First>` und `<first>` verwenden!

## Regel 6: Text-Formatierungen

- **Beispiel:** `<P>`This is a paragraph. It has a whole bunch of space.`</P>`
- HTML reduziert Text-Formatierungen (*white spaces*) auf ein Leerzeichen :



## Regel 6: White Space

- **XML belässt alle Formatierungen im Text.**
- **Beispiel:** Der Inhalt von

```
<P>This is a paragraph. It has a whole bunch  
of space.</P>
```

ist also:

```
This is a paragraph. It has a whole bunch  
of space.
```

- **Beachte:** Von Browsern werden die Formatierungen allerdings *nicht* angezeigt.
- **Grund:** XML-Dokumente werden zur Darstellung im Browser in HTML umgewandelt.

## XML-Editoren

- XML-Dokumente werden normalerweise mit speziellen Editoren erstellt und modifiziert.
- Meistbenutzte XML-Editor ist XML Spy.
- steht in den PC-Pools zur Verfügung
- gibt es aber auch als kostenlose vierwöchige Testlizenz  
→ [www.xmlspy.com](http://www.xmlspy.com)



## Kleiner Online-Test zu XML



- [http://www.w3schools.com/xml/xml\\_quiz.asp](http://www.w3schools.com/xml/xml_quiz.asp)
- W3 Schools bietet auch Online-Tutorials zu XML-Technologien an.

## Namensräume

## Namenskonflikte

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<person>
  <name>
    <title>Sir</title>
    <first>John</first>
    <middle> Fitzgerald Johansen</middle>
    <last>Doe</last>
  </name>
  <position>Vice President of
Marketing</position>
  <résumé>
    <html>
      <title>Resume of John Doe</title>
      <head>
        <body>
          <h1>John Doe</h1>
          <p>John's a great guy, you know?</p>
        </body>
      </html>
    </résumé>
  </person>
```

- **Namenskonflikt:** gleicher Name, aber unterschiedliche Bedeutung
- z.B. Titel einer Person vs. Titel eines Dokumentes
- in einem Dokument unterschiedliche Vokabularien

© Klaus Schild, 2004

37

## Präfixe

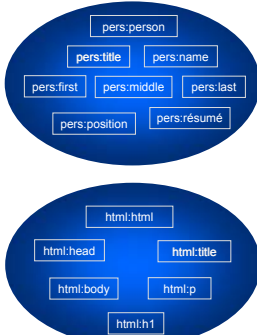
```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<pers:person>
  <pers:age>
    <pers:title>Sir</pers:title>
    <pers:fn1>John</pers:fn1>
    <pers:middle> Fitzgerald Johansen</pers:middle>
    <pers:last>Doe</pers:last>
  </pers:person>
  <pers:position>Vice President of Marketing</pers:position>
  <pers:résumé>
    <html:head>
      <html:title>Resume of John Doe</html:title>
      <html:head>
        <html:body>
          <html:h1>John Doe</html:h1>
          <html:p>John's a great guy, you know?</html:p>
        </html:body>
      </html:head>
    </pers:résumé>
  </pers:person>
```

- Präfixe geben den Kontext an: Aus welchem Bereich stammt der Name?
- z.B. pers:title vs. html:title
- Auf diese Weise werden auch Namenskonflikte in Programmiersprachen aufgelöst:  
Z.B. java.applet.Applet

© Klaus Schild, 2004

38

## Namensräume



- **Namensraum (namespace):** Alle Bezeichner (Namen) mit identischen Anwendungskontext
- Namensräume müssen eindeutig identifizierbar sein.

© Klaus Schild, 2004

39

## Namensräume in XML

- Im WWW müssen Namensräume *global* eindeutig sein.
- In XML wird ein Namensraum deshalb mit einer URI identifiziert
- Zuerst wird einem Präfix ein bestimmter Namensraum zugeordnet, z.B.:

```
xmlns:html="http://www.w3.org/1999/xhtml"
```

Namensraum-Präfix

Namensraum-Bezeichner (URI)

- Anschließend kann der Namensraum-Präfix einem Namen vorangestellt werden: z.B. html:title
- **Beachte:** Wahl des Präfixes egal

© Klaus Schild, 2004

40

## Beispiel

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<pers:person xmlns:pers="http://semafema.com/pers"
xmlns:html="http://www.w3.org/1999/xhtml">
  <pers:age>
    <pers:title>Sir</pers:title>
    <pers:fn1>John</pers:fn1>
    <pers:middle> Fitzgerald Johansen</pers:middle>
    <pers:last>Doe</pers:last>
  </pers:age>
  <pers:position>Vice President of Marketing</pers:position>
  <pers:résumé>
    <html:head>
      <html:title>Resume of John Doe</html:title>
    </html:head>
    <html:body>
      <html:h1>John Doe</html:h1>
      <html:p>John's a great guy, you know?</html:p>
    </html:body>
  </pers:résumé>
</pers:person>
```

**Beachte:** In einem Element können auch *mehrere* Namensraum-Präfixe definiert werden.

© Klaus Schild, 2004

41

## Uniform Resource Identifier (URI)

- eindeutige Bezeichner für Ressourcen im WWW
- Eine URI kann den physischen Aufenthaltsort einer Resource beschreiben:  
http://www.w3.org/1999/xhtml
- Solche URIs werden auch Uniform Resource Locations (URLs) genannt.

© Klaus Schild, 2004

42

## Uniform Resource Identifier (URI)



- Eine URI kann auch ein Namen einer Resource unabhängig von deren physischen Aufenthaltsort sein:  
um:oasis:names:specification:docbook:dtd:xml:4.1.2  
um:isbn:1-861005-59-8
- um:oasis und um:isbn werden URI-Schemata (*URI schemes*) genannt.
- URI-Schemata können bei der IANA registriert werden:
  - genaue Festlegung der Syntax
  - Wer vergibt die dazugehörigen Namen?



© Klaus Schild, 2004

43

## URIs als Namensraumbezeichner



- URI des Namensraumes *kann* (muss aber nicht) eine Beschreibung des Namensraumes enthalten.
- Die URI muss nicht einmal existieren!
- Entscheidend ist, dass bei tatsächlich existierenden URIs Eindeutigkeit sichergestellt ist.

© Klaus Schild, 2004

44

## Standard-Namensräume



```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<person xmlns="http://www.w3.org/1999/xhtml">
  <name>
    <title>Sir</title>
    <first>John</first>
    <middle>Fitzgerald Johansen</middle>
    <last>Doe</last>
  </name>
  <position>Vice President of Marketing</position>
  <resume>
    <html:html>
      <html:head>
        <html:title>Resume of John Doe</html:title>
      </html:head>
      <html:body>
        <html:h1>John Doe</html:h1>
        <html:p>John's a great guy, you know?
      </html:p>
      </html:body>
    </html:html>
  </resume>
</person>
```

- Für jedes Element kann ein Namensraum als Standard (*default*) festgelegt werden, z.B.:

xmlns="http://www.w3..."

- Element-Namen *ohne* Präfix gehören dann zum Standard-Namensraum.
- Kind-Elemente erben Standard-Namensraum vom Eltern-Element.

- **Beachte:** Standard-Namensraum gilt hier auch für person.

© Klaus Schild, 2004

45

## Wo Namensräume definieren?



```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<person xmlns="http://www.w3.org/1999/xhtml">
  <name>
    <title>Sir</title>
    <first>John</first>
    <middle>Fitzgerald Johansen</middle>
    <last>Doe</last>
  </name>
  <position>Vice President of Marketing</position>
  <resume>
    <html:html>
      <html:head>
        <html:title>Resume of John Doe</html:title>
      </html:head>
      <html:body>
        <html:h1>John Doe</html:h1>
        <html:p>John's a great guy, you know?
      </html:p>
      </html:body>
    </html:html>
  </resume>
</person>
```

- Namensraum-Präfixe und Standard-Namensräume müssen nicht im Wurzel-Element definiert werden.
- am besten dort definieren, wo sie benutzt werden

© Klaus Schild, 2004

46

## Gültigkeitsbereich



```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<person xmlns="http://www.w3.org/1999/xhtml">
  <name>
    <title>Sir</title>
    <first>John</first>
    <middle>Fitzgerald Johansen</middle>
    <last>Doe</last>
  </name>
  <position>Vice President of Marketing</position>
  <resume xmlns="http://www.w3.org/1999/xhtml">
    <html:html>
      <html:head>
        <html:title>Resume of John Doe</html:title>
      </html:head>
      <html:body>
        <html:h1>John Doe</html:h1>
        <html:p>John's a great guy, you know?
      </html:p>
      </html:body>
    </html:html>
  </resume>
</person>
```

- Ein Kind-Element erbt alle Namensraum-Definitionen seines Eltern-Elementes.
- Ererbter Standard-Namensraum kann überschrieben werden.

© Klaus Schild, 2004

47

## Qualified vs. Unqualified



- Ein Name heißt **namensraumeingeschränkt** (*qualified*), wenn er einem Namensraum zugeordnet ist.
- Ist gibt zwei Möglichkeiten, diese Zuordnung vorzunehmen:
  1. Standard-Namensraum festlegen
  2. Namensraum-Präfix voranstellen

© Klaus Schild, 2004

48

## Beispiel 1

```
<?xml version="1.0"?>
<BookStore xmlns="http://www.books.org">
  <Book>
    <Title>My Life and Times</Title>
    <Author>Paul McCartney</Author>
    <Date>July, 1998</Date>
    <ISBN>94303-12021-43892</ISBN>
    <Publisher>McMillin Publishing</Publisher>
  </Book>
</BookStore>
```

- alle Element-Namen (einschl. BookStore!) Teil des Standard-Namensraumes
- alle Element-Namen daher namensraumeingeschränkt (*qualified*)

© Klaus Schild, 2004

49

## Beispiel 2

```
<?xml version="1.0"?>
<bk:BookStore xmlns:bk="http://www.books.org">
  <bk:Book>
    <bk:Title>My Life and Times</bk:Title>
    <bk:Author>Paul McCartney</bk:Author>
    <bk:Date>July, 1998</bk:Date>
    <bk:ISBN>94303-12021-43892</bk:ISBN>
    <bk:Publisher>McMillin Publishing</bk:Publisher>
  </bk:Book>
</bk:BookStore>
```

- alle Element-Namen haben Namensraum-Präfix
- alle Element-Namen daher namensraumeingeschränkt (*qualified*).

© Klaus Schild, 2004

50

## Beispiel 3

```
<?xml version="1.0"?>
<bk:BookStore xmlns:bk="http://www.books.org">
  <Book>
    <Title>My Life and Times</Title>
    <Author>Paul McCartney</Author>
    <Date>July, 1998</Date>
    <ISBN>94303-12021-43892</ISBN>
    <Publisher>McMillin Publishing</Publisher>
  </Book>
</bk:BookStore>
```

unqualified

- kein Standard-Namensraum festgelegt
- Element-Namen *ohne* Namensraum-Präfix daher *keinem* Namensraum zugeordnet (*unqualified*)

© Klaus Schild, 2004

51

## Namensräume für Attribute

```
<html xmlns:1 "http://www.w3.org/1999/xhtml" ...>
  <person xmlns="http://www.ferret.com/pers"
    name="J" ...>
    <title>Johannes</title>
    <first>Johannes</first>
    <middle>FRitzgerald Johannes</middle>
    <last>Doe</last>
    </name>
    <position>Vice President of Marketing</position>
  </person>
  <html:html
    xmlns:html="http://www.w3.org/1999/xhtml" ...>
    <html:head>
      <html:title>Resume of John Doe</html:title>
    </html:head>
    <html:body>
      <html:h1>John Doe</html:h1>
      <html:p font-family="FONT-FAMILY: Arial" style="color:
        a great gray, you know!">John's
    </html:body>
  </html:html>
</html>
```

- Beachte:** Attribute gehören *nicht* automatisch zum Standard-Namensraum.
- Grund:** Attribute wie *id* werden in verschiedenen Elementen (und Namensräumen) verwendet, sollten sich aber nicht unbedingt unterscheiden.
- Attribut-Namen kann aber ein Namensraum-Präfix vorangestellt werden.

© Klaus Schild, 2004

52

## Namensräume & Semantik

- Bedeutung von `<p>Text</p>`
  - für HTML: festgelegt
  - für XML: offen
- In XML können Namensräume Bedeutung festlegen.
- HTML auf bestimmte Semantik festgelegt, XML nicht.

<http://www.w3.org/1999/xhtml>



```
<xhtml:p xmlns:xhtml="http://www.w3.org/1999/xhtml">Text</xhtml:p>
```

© Klaus Schild, 2004

53

## Und das war es schon?

- Ja!
- Syntax wohlgeformter XML-Dokumente wurde vollständig vorgestellt.
- einzige Ausnahme: Prozessorinstruktionen
- XML-Syntax ist also sehr einfach.
- Gleichzeitig ist XML aber erweiterbar.
- Das ist genau die Stärke von XML: einfach und flexibel!

© Klaus Schild, 2004

54

## Wie geht es weiter?



- Syntax wohlgeformter XML-Dokumente
- Namensräume zur Auflösung von Namenskonflikten
- Festlegung der Semantik von XML-Elementen
  - Beschreibung von Klassen von Dokumenten mit DTDs und XML-Schema