

Musterlösung des Übungsblattes 1

Wie kann Wohlgeformtheit bzgl. XML 1.0 und 1.1 erreicht werden?

- streng genommen gar nicht!
- Grund: XML-Deklaration
- entweder nicht vorhanden → nicht wohlgeformt bzgl. XML 1.1
- oder vorhanden, dann
 - entweder version="1.0" → nicht wohlgeformt bzgl. XML 1.1
 - oder version="1.1" → nicht wohlgeformt bzgl. XML 1.0

Was kann erreicht werden?

Sowohl XML-1.0-Parser als auch XML-1.1-Parser können das Dokument verarbeiten.

Und wie?

- keine XML-Deklaration oder eine mit version="1.0"
- XML-1.0-Parser
- XML-1.1-Parser:
 - XML 1.1 verlangt von Parseern, dass beide Versionen erkannt werden:
 - wenn keine XML-Deklaration oder explizit Version 1.0: Wohlgeformtheit gemäß XML 1.0
 - in allen anderen Fällen: Wohlgeformtheit gemäß XML 1.1

```
1      <?xml version="1.0"?>
2      = <ipo:purchaseOrder xmlns:xsi="
      http://www.w3.org/2001/XMLSchema-instance" xmlns:ipo="
      http://www.altova.com/IPO" orderDate="1999-12-01">
3      ⊕ <ipo:shipTo export-code="1" xsi:type="ipo:EU-Address">
9      ⊕ <ipo:billTo xsi:type="ipo:US-Address">
16     ⊕ <ipo:Items>
57     </ipo:purchaseOrder>
```

Musterlösung mit Standardnamensraum

```
1 <?xml version="1.0"?>
2 <purchaseOrder xmlns:xsi="
  http://www.w3.org/2001/XMLSchema-instance" xmlns="
  http://www.altova.com/IPO" xmlns:ipo="
  http://www.altova.com/IPO" orderDate="1999-12-01">
3   <shipTo export-code="1" xsi:type="ipo:EU-Address">
9   <billTo xsi:type="ipo:US-Address">
16  <Items>
57 </purchaseOrder>
```

```
1 <?xml version="1.0"?>
2 <purchaseOrder xmlns:xsi="
  http://www.w3.org/2001/XMLSchema-instance" xmlns="
  http://www.altova.com/IPO" xmlns:ipo="
  http://www.altova.com/IPO" orderDate="1999-12-01">
3   <shipTo export-code="1" xsi:type="ipo:EU-Address">
4     <name>Helen Zoe</name>
5     <street>47 Eden Street</street>
6     <city>Cambridge</city>
7     <postcode>126</postcode>
8   </shipTo>
9   <billTo xsi:type="ipo:US-Address">
16  <Items>
57 </purchaseOrder>
```

```
1      <?xml version="1.0"?>
2      <purchaseOrder xmlns:xsi="
      http://www.w3.org/2001/XMLSchema-instance" xmlns="
      http://www.altova.com/IPO" xmlns:ipo="
      http://www.altova.com/IPO" orderDate="1999-12-01">
3      <shipTo export-code="1" xsi:type="ipo:EU-Address">
9      <billTo xsi:type="ipo:US-Address">
10     <name>Robert Smith</name>
11     <street>8 Oak Avenue</street>
12     <city>Old Town</city>
13     <state>AK</state>
14     <zip>95819</zip>
15     </billTo>
16     <Items>
57     </purchaseOrder>
```

```
1 <?xml version="1.0"?>
2 <purchaseOrder xmlns:xsi="
  http://www.w3.org/2001/XMLSchema-instance" xmlns="
  http://www.altova.com/IPO" xmlns:ipo="
  http://www.altova.com/IPO" orderDate="1999-12-01">
3 <shipTo export-code="1" xsi:type="ipo:EU-Address">
9 <billTo xsi:type="ipo:US-Address">
16 <Items>
17 <item partNum="833-AA">
18 <productName>Lapis necklace</productName>
19 <quantity>2</quantity>
20 <price>99.95</price>
21 <comment>Need this for the holidays!</comment>
22 <shipDate>1999-12-05</shipDate>
23 </item>
24 <item partNum="748-OT">
31 <item partNum="783-KL">
37 <item partNum="238-KK">
44 <item partNum="229-OB">
50 <item partNum="128-UL">
56 </Items>
57 </purchaseOrder>
```

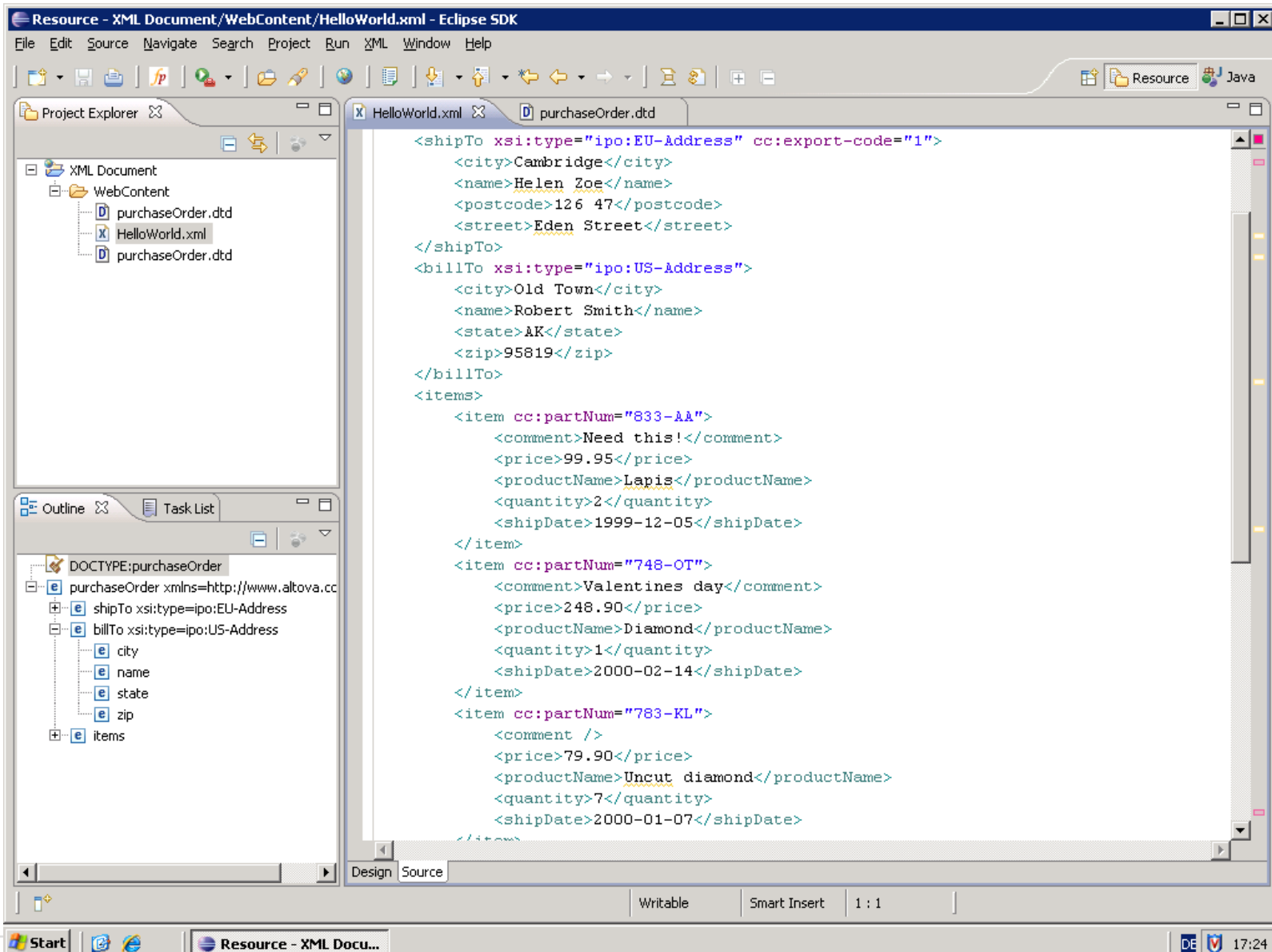

- **xmlns="URI"** statt xmlns:prefix="URI,,
- Namensraum-Präfix kann weggelassen werden.
- Standard-Namensraum gilt für das Element, wo er definiert ist.
- Kind-Elemente erben Standard-Namensraum von ihrem Eltern-Element.
- Ausnahme: Standard-Namensraum wird überschrieben
- Beachte: Standardnamensräume gelten nicht für Attribute

Qualified vs. Unqualified

- Element- oder Attribut-Name heißt namensraumeingeschränkt (qualified), wenn er einem Namensraum zugeordnet ist.
- Einschränkung vom Element-Namensraum:
 1. Standard-Namensraum festlegen
 2. Namensraum-Präfix voranstellen
- Einschränkung vom Attribut-Namensraum:
 1. Namensraum-Präfix voranstellen

- Attribute in XML sind sogenannte *assoziierte Knoten*. Sie werden nicht wie normale Kindelemente eines Elements behandelt. Bei Namensräumen heißt das, dass Attribute nicht im Namensraum des Elements stehen, in dem sie notiert sind, sondern vorgabemäßig im Null-Namensraum.
- Es gibt Fälle, in denen man dies explizit ändern will. In diesem Fall müssen die einzelnen Attribute mit einem Präfix versehen werden:
- `<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"> <a xlink:href="grafik2.svg">Link zu Grafik 2 </svg>`
- Es gibt für Attribute keine Möglichkeit, sie ohne Präfix in einen bestimmten Namensraum zu setzen.

Musterlösung (Teil 2)






The screenshot shows the Eclipse IDE interface with the following components:

- Project Explorer:** Shows a project named 'XML Document' containing a 'WebContent' folder. Inside 'WebContent', there are two 'purchaseOrder.dtd' files and one 'HelloWorld.xml' file.
- Outline:** Shows the DOCTYPE:purchaseOrder structure with elements: purchaseOrder (xmlns=http://www.altova.cc), shipTo (xsi:type=ipo:EU-Address), billTo (xsi:type=ipo:US-Address), city, name, state, zip, and items.
- Source Editor:** Displays the XML content of 'HelloWorld.xml':

```
<shipTo xsi:type="ipo:EU-Address" cc:export-code="1">
  <city>Cambridge</city>
  <name>Helen Zoe</name>
  <postcode>126 47</postcode>
  <street>Eden Street</street>
</shipTo>
<billTo xsi:type="ipo:US-Address">
  <city>Old Town</city>
  <name>Robert Smith</name>
  <state>AK</state>
  <zip>95819</zip>
</billTo>
<items>
  <item cc:partNum="833-AA">
    <comment>Need this!</comment>
    <price>99.95</price>
    <productName>Lapis</productName>
    <quantity>2</quantity>
    <shipDate>1999-12-05</shipDate>
  </item>
  <item cc:partNum="748-OT">
    <comment>Valentines day</comment>
    <price>248.90</price>
    <productName>Diamond</productName>
    <quantity>1</quantity>
    <shipDate>2000-02-14</shipDate>
  </item>
  <item cc:partNum="783-KL">
    <comment />
    <price>79.90</price>
    <productName>Uncut diamond</productName>
    <quantity>7</quantity>
    <shipDate>2000-01-07</shipDate>
  </item>
</items>
```
- Task List:** Empty.
- Bottom Bar:** Shows 'Design' and 'Source' tabs, 'Writable', 'Smart Insert', and '1 : 1'.

Musterlösung des Übungsblattes 2

ipoDefaultNS.xml: purchaseOrder

```
1      <?xml version="1.0"?>
2       <purchaseOrder xmlns:xsi="
      http://www.w3.org/2001/XMLSchema-instance" xmlns="
      http://www.altova.com/IPO" xmlns:ipo="
      http://www.altova.com/IPO" orderDate="1999-12-01">
3       <shipTo export-code="1" xsi:type="ipo:EU-Address">
9       <billTo xsi:type="ipo:US-Address">
16      <Items>
57     </purchaseOrder>
```

Deklaration von purchaseOrder

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT purchaseOrder (shipTo, billTo, Items)>
<!ATTLIST purchaseOrder
  xmlns:xsi CDATA #FIXED "http://www.w3.org/2001/XMLSchema-instance"
  xmlns CDATA #FIXED "http://www.altova.com/IPO"
  xmlns:ipo CDATA #FIXED "http://www.altova.com/IPO"
  orderDate CDATA #REQUIRED >
```

Probleme:

- funktioniert nur mit Präfixen xsi und ipo
- Standard-Namensraum vorgeschrieben (ipo.xml ohne Standardnamensraum wäre also nicht gültig)
- Struktur von orderDate nicht eingeschränkt „fffzzz“???

ipoDefaultNS.xml: shipTo

```
1      <?xml version="1.0"?>
2      = <purchaseOrder xmlns:xsi="
      http://www.w3.org/2001/XMLSchema-instance" xmlns="
      http://www.altova.com/IPO" xmlns:ipo="
      http://www.altova.com/IPO" orderDate="1999-12-01">
3      ○ <shipTo export-code="1" xsi:type="ipo:EU-Address">
4          <name>Helen Zoe</name>
5          <street>47 Eden Street</street>
6          <city>Cambridge</city>
7          <postcode>126</postcode>
8      </shipTo>
9      ⊕ <billTo xsi:type="ipo:US-Address">
16     ⊕ <Items>
57     </purchaseOrder>
```


Deklaration von shipTo

```
<!ELEMENT shipTo (name, street, city, (postcode | (state, zip)))>
<!ATTLIST shipTo
    export-code CDATA #IMPLIED
    xsi:type (ipo:EU-Address | ipo:US-Address) #REQUIRED
>
```

- In XML selbst ist es erlaubt, Elemente zu definieren, die den Doppelpunkt bereits im Elementnamen tragen. (nicht empfehlenswert)
- Dateien, die Namensräume verwenden, sind dadurch wohlgeformte XML-Dokumente.

Deklaration von shipTo

```
<!ELEMENT shipTo (name, street, city, (postcode | (state, zip)))>  
<!ATTLIST shipTo  
    export-code CDATA #IMPLIED  
    xsi:type (ipo:EU-Address | ipo:US-Address) #REQUIRED  
>
```

Probleme:

- kein Zusammenhang zwischen Struktur von shipTo und xsi:type
- kein Zusammenhang zwischen xsi:type und export-code

ipoDefaultNS.xml: billTo

```
1      <?xml version="1.0"?>
2      <purchaseOrder xmlns:xsi="
      http://www.w3.org/2001/XMLSchema-instance" xmlns="
      http://www.altova.com/IPO" xmlns:ipo="
      http://www.altova.com/IPO" orderDate="1999-12-01">
3      <shipTo export-code="1" xsi:type="ipo:EU-Address">
9      <billTo xsi:type="ipo:US-Address">
10         <name>Robert Smith</name>
11         <street>8 Oak Avenue</street>
12         <city>Old Town</city>
13         <state>AK</state>
14         <zip>95819</zip>
15     </billTo>
16     <Items>
57 </purchaseOrder>
```

Deklaration von billTo

```
<!ELEMENT billTo (name, street, city, (postcode | (state, zip)))>  
<!ATTLIST billTo  
    xsi:type (ipo:EU-Address | ipo:US-Address) #REQUIRED  
>
```

Problem:

- kein Zusammenhang zwischen Struktur von billTo und xsi:type

ipoDefaultNS.xml: Items und item

```
1      <?xml version="1.0"?>
2      □ <purchaseOrder xmlns:xsi="
      http://www.w3.org/2001/XMLSchema-instance" xmlns="
      http://www.altova.com/IPO" xmlns:ipo="
      http://www.altova.com/IPO" orderDate="1999-12-01">
3      ⊕ <shipTo export-code="1" xsi:type="ipo:EU-Address">
9      ⊕ <billTo xsi:type="ipo:US-Address">
16     ⊖ <Items>
17     ⊖   <item partNum="833-AA">
18         <productName>Lapis necklace</productName>
19         <quantity>2</quantity>
20         <price>99.95</price>
21         <comment>Need this for the holidays!</comment>
22         <shipDate>1999-12-05</shipDate>
23     ⊖   </item>
24     ⊕   <item partNum="748-OT">
31     ⊕   <item partNum="783-KL">
37     ⊕   <item partNum="238-KK">
44     ⊕   <item partNum="229-OB">
50     ⊕   <item partNum="128-UL">
56     ⊖   </Items>
57     </purchaseOrder>
```

```
<!ELEMENT Items (item+)>
```

```
<!ELEMENT item (productName, quantity, price, comment?, shipDate)>
```

```
<!ATTLIST item partNum CDATA #REQUIRED>
```

oder besser

```
<!ATTLIST item partNum ID #REQUIRED>
```

XML sieht eine Möglichkeit vor, dem Parser mitzuteilen, dass der zugewiesene Wert eines bestimmten Attributs dokumentweit nur einmal vorkommen darf. Dies ist ein wichtiges Feature vor allem im Hinblick auf Script-Sprachen. Denn nur bei dokumentweit eindeutigen, identifizierenden Werten ist es möglich, ein Element über den Identifikationswert anzusprechen.

Die Wertzuweisungen an ein Attribut vom Typ ID müssen den Regeln für Namen entsprechen! (Darf also nicht mit einer Ziffer beginnen)

Siehe auch **IDREF**

```
<!ELEMENT name (#PCDATA)>  
<!ELEMENT street (#PCDATA)>  
<!ELEMENT city (#PCDATA)>  
<!ELEMENT postcode (#PCDATA)>  
<!ELEMENT state (#PCDATA)>  
<!ELEMENT zip (#PCDATA)>  
<!ELEMENT productName (#PCDATA)>  
<!ELEMENT quantity (#PCDATA)>  
<!ELEMENT price (#PCDATA)>  
<!ELEMENT comment (#PCDATA)>  
<!ELEMENT shipDate (#PCDATA)>
```

Deklaration von shipTo und billTo

```
<!ELEMENT shipTo (name, street, city, (postcode | (state, zip)))>  
<!ATTLIST shipTo  
    export-code CDATA #IMPLIED  
    xsi:type (ipo:EU-Address | ipo:US-Address) #REQUIRED  
>
```

```
<!ELEMENT billTo (name, street, city, (postcode | (state, zip)))>  
<!ATTLIST billTo  
    xsi:type (ipo:EU-Address | ipo:US-Address) #REQUIRED  
>
```

- an zwei Stellen identische Strukturen
⇒ nicht veränderungsfreundlich

Verbesserte Version

```
<!ENTITY % Address "(name, street, city,(postcode|(state, zip)))">  
<!ELEMENT shipTo %Address;>  
<!ELEMENT billTo %Address;>
```

⇒ wird Parameter Entity genannt

Musterfragen

```
<xs:element name="ethnos">  
  <xs:complexType>  
    <xs:attribute name="self" type="xs:string" use="required"/>  
  </xs:complexType>  
</xs:element>
```

To which of the following DTD declarations is the "self" attribute above equivalent?

1. `<!ATTLIST ethnos self (#PCDATA) #REQUIRED>`
2. `<!ATTLIST ethnos self PCDATA #REQUIRED>`
3. `<!ATTLIST ethnos self CDATA #FIXED>`
4. `<!ATTLIST ethnos self PCDATA #IMPLIED>`
5. `<!ATTLIST ethnos self CDATA #REQUIRED>`

Was könnte gültig sein?

A. `<?xml version="1.0"?>`

`<!DOCTYPE galaxy SYSTEM "cosmos.dtd">`

`<galaxy>M85</galaxy>`

B. `<?xml version="1.0" encoding="UTF-8" ?>`

`<!DOCTYPE galaxy [<!ELEMENT galaxy (#PCDATA)>]>`

`<galaxy>M85</galaxy>`

C. `<?xml version="1.0"?>`

`<!DOCTYPE SYSTEM "http://www.example.com/cosmos.dtd">`

`<galaxy>M85</galaxy>`

Was könnte gültig sein?

D. `<?xml version="1.0">`

```
<!DOCTYPE galaxy SYSTEM
    "http://www.example.com/cosmos.dtd">
```

```
<galaxy>M85</galaxy>
```

E. `<?xml version="1.0"?>`

```
<!DOCTYPE PUBLIC "http://www.example.com/cosmos.dtd">
```

```
<galaxy>M85</galaxy>
```

Entity Declarations

```
<!ENTITY species "erectus">  
<!ENTITY species "habilis">  
<!ENTITY species "sapiens">  
<!ENTITY species "">
```

What value will the expression `&species;` yield in an instance document based on this DTD?

1. erectus
2. habilis
3. sapiens
4. (empty)
5. The fragment is not valid DTD

```
<!ATTLIST x y ENTITY #REQUIRED>
```

According to this definition:

1. The name of the *y* element must match the name of an unparsed entity defined in the DTD.
2. The value of the *y* element must match the name of an unparsed entity defined in the DTD.
3. The name of the *y* attribute must match the name of an unparsed entity defined in the DTD.
4. The value of the *y* attribute must match the name of an unparsed entity defined in the DTD.
5. None of the above.

Element-Deklarationen

```
<!ELEMENT train (engine?, wagon*)>  
<!ELEMENT engine (#PCDATA)>  
<!ELEMENT wagon (#PCDATA)>
```

Which XML would be successfully validated?

A. `<train />`

B. `<train>`

`<engine> That Could </engine>`

`</train>`


```
<!ELEMENT train (engine?, wagon*)>  
<!ELEMENT engine (#PCDATA)>  
<!ELEMENT wagon (#PCDATA)>
```

Which XML would be successfully validated?

C. <train>

<engine> That Could

<wagon> Caboose </wagon>

</engine>

</train>

Element-Deklarationen

```
<!ELEMENT train (engine?, wagon*)>  
<!ELEMENT engine (#PCDATA)>  
<!ELEMENT wagon (#PCDATA)>
```

Which XML would be successfully validated?

D. <train>

<engine> No. 7 </engine>

<wagon> Pullman </wagon>

<wagon> Diner </wagon>

</train>

Element-Deklarationen

```
<!ELEMENT train (engine?, wagon*)>  
<!ELEMENT engine (#PCDATA)>  
<!ELEMENT wagon (#PCDATA)>
```

Which XML would be successfully validated?

E. <train>

<engine> No. 8 </engine>

<engine> No. 9 </engine>

<wagon> Wagon-Lits </wagon>

</train>

Was wäre gültig?

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE audio_presentation [
  <!ENTITY data "c:.wav">
  <!ELEMENT audio_presentation (#PCDATA)>
  <!NOTATION au PUBLIC "au 1.0" "http:\\www.example.com.exe">
  <!NOTATION wav PUBLIC "wav 1.0" "http:\\www.example.com.exe">
  <!ATTLIST audio_presentation modality NOTATION (au|wav) #REQUIRED>
]>
```

- A. `<audio_presentation modality="au|wav">&data;</audio_presentation>`
- B. `<audio_presentation modality="wav">&data;</audio_presentation>`
- C. `<audio_presentation>&data;</audio_presentation>`
- D. `<audio_presentation modality="data"></audio_presentation>`
- E. `<audio_presentation modality="&data;">wav</audio_presentation>`

Was wäre gültig?

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE audio_presentation [
  <!ENTITY data "c:.wav">
  <!ELEMENT audio_presentation (#PCDATA)>
  <!NOTATION au PUBLIC "au 1.0" "http:\\www.example.com.exe">
  <!NOTATION wav PUBLIC "wav 1.0" "http:\\www.example.com.exe">
  <!ATTLIST audio_presentation modality NOTATION (au|wav) #REQUIRED>
]>
```

- Notationen sind "Verarbeitungshinweise" an die interpretierende Software, wenn Sie externe Daten in XML einbinden, also etwa Grafiken, Multimedia, Java-Applets, Flash-Filme oder dergleichen. Solche Daten werden vom XML-Parser nicht direkt verarbeitet. Mit Hilfe der Notationen steht jedoch eine Möglichkeit zur Verfügung, der XML-verarbeitenden Software Details über die referenzierten Daten mitzuteilen.

-Sinnvolle Anwendungsfälle von XML

-Was kann man mit einer **DTD** machen?

-Was geht mit einer DTD nicht?

-Wann DTD einsetzen statt **XML Schema**?

-**Namensräume**: wozu?

-Default-Namensräume, Qualified/unqualified (speziell Attribute)

-**Regeln für Namen** in XML

-Abstraktion von Textvorlage