



*XML Pointer Language (XPointer),  
XML Linking Language (XLink)*

<http://www.w3.org/TR/xpointer>

<http://www.w3.org/TR/xlink>

<http://www.w3.org/TR/xbase>



*XML Pointer Language*

Übersicht, Ursprünge

Konzepte

Erweiterungen

# XPointer

- **Was ist XPointer?**
  - Die Spezifikation zur XML-konformen Formulierung von Verweisen auf Dokumente, Dokumentteile und –bereiche.
  - Ein Hilfsstandard, der XLink zuarbeitet und auf XPath aufbaut
  - Eine Verallgemeinerung der *URI* und *URI references* aus HTML.
  - Eine W3C-Empfehlung
- **Was sind die Ursprünge von XPointer?**
  - HTML, insbesondere das Element `<a>` und das URL/URI-Konzept
  - HyTime (ISO/IEC 10744) – definiert Datentypen für Ortsspezifikationen für zahlreiche Arten von Daten.
  - TEI-Richtlinien (TEI=*Text Encoding Initiative*) für eine formale Syntax zu verallgemeinerten Zeigern, insb. im Markup-Kontext.
- **Warum reicht XPath nicht?**
  - XPath-Ausdrücke müssen „verpackt“ werden, um gültige Links zu ergeben
  - XPointer erweitert XPath um Referenzierungskonzepte:
    - Orts- und Bereichsangaben

# XPointer

- **URI-Referenzen und XPath**
  - URI-Referenzen bestehen aus
    - dem URI-Teil
    - dem Trennzeichen # sowie
    - dem *fragment identifier*.
  - Beispiel (farbcodiert: **URI**, Trennzeichen, **fragment identifier**):  
`http://www.myorg.com/dir/foo.html#toc`
- **HTML-Vergleich, mit „Anker“-Element `<a>`**
  - Anker für *fragment identifier* (*hierhin zeigt der o.g. Verweis*):  
`<h2><a name="toc">Table of Content</a>`
  - Unterscheide - Anker für ein Link:  
`<a href="./bar.html"> Click here</a> to continue.`  
(Auszüge aus dem hypothetischen „foo.html“)

# XPointer

- **URI mit XPointer**

- Der XPointer-Standard ersetzt die *fragment identifier* in URI-Referenzen durch Ausdrücke der Form

...#xpointer( XPath\_expression )

- Beispiel:

```
http://www.myorg.com/foo.xml#  
xpointer(//table-of-content[1])
```

- Allgemein:

- Hinter dem Trennzeichen # dürfen auch mehrere xpointer-Anweisungen stehen, optional durch *white space* getrennt („XPointer-Teile“).
- Dieses Verfahren# kann z.B. hilfreich sein, um Ersatzorte anzugeben, falls die primäre Referenz nicht mehr besteht.

# XPointer

- **URI mit XPointer: Namespace-Referenzen**

- Präfixwerte für Namensräume werden auch in XPointer verwendet (manchmal unvermeidlich).
- Die Deklaration dieser Namensräume wird der XPointer-“Funktion“ vorangestellt, etwa wie in:

```
http://www.myorg.com/foo.xml#  
xmlns(pfx=http://www.meinbeispiel.de/ns/a1)  
xpointer(//pfx:toc)
```

- Im Dokument selbst kann (derselbe!) Namensraum dabei über ein anderes Präfix verwaltet sein:

```
<my:doc  
  xmlns:my="http://www.meinbeispiel.de/ns/a1">...  
  <my:toc>Table of Content</my:toc>  
</my:doc>
```

# XPointer

- **Neue Möglichkeiten durch XPointer**
  - Verweise auf präzise Stellen in XML-Dokumenten ganz ohne „Anker“
  - Bei Bedarf zeichengenaue Positionierungen
  - Bequeme Positionierungen mittels „id()“-Funktion
- **Neues Konzept: *location***
  - Punkte:
    - Präzise Ortsangaben im Datenmodell: vor, in bzw. hinter bestimmten Knoten des Datenbaums
  - Bereiche:
    - Knoten (in Dokumentenreihenfolge) zwischen Punkten.

# XPointer

- **Datenmodell**
  - Das Datenmodell von XPointer kommt dem XML Infoset näher als das von XPath, denn einzelne Zeichen sind adressierbar.
  - Ähnlich wie XSLT gestattet auch XPointer die Anwendung auf Dokumentfragmente, wie sie z.B. in *external entities* erscheinen. *Root*-Elemente dürfen deshalb Textknoten und mehr als einen Kindelementknoten besitzen!
- **XPointer-Erweiterungen zu XPath**
  - XPath-Ausdrücke in XPointer führen nicht immer zu sinnvollen Verweisen. Während XPath-Ausdrücke auch leere Mengen liefern können, führt das bei XPointer zu Fehlern.
  - XPointer führt zwei **Kurzschreibweisen** ein.
  - **Neue Funktionen** unterstützen das *location*-Konzept

## XPointer-Kurzschreibweisen

- *bare names*:
  - Speziell für die Verwendung von `id()` gilt:  
Statt `...#xpointer( id("ID12345") )`  
kann man auch nur die ID (ohne " ") schreiben:  
`...#ID12345`
- *child sequences*:
  - Elemente im Dokumentenbaum können auch durch eine Art Index angesprochen werden:  
`...#ID12345/1/2`
  - Bedeutung: Das zweite Unterelement des ersten Unterelements des Elements mit der ID „ID12345“

## XPointer: Punkte und Bereiche

- **Der Punktbeff**
  - Um zeichengenau adressieren zu können, definiert XPointer zwei neue Typen von Positionsangaben, *point* und *range*.
  - XPath-Ausdrücke lassen sich in XPointer auch auf diese Datentypen anwenden.
  - Ein Punkt besteht aus
    - seinem *Container-Knoten* und
    - einer nicht-negativen ganzen Zahl (*Index*)
  - Fall „Knoten kann Kindknoten haben“ (*node point*):
    - Der Index adressiert den jeweiligen Kindknoten
    - 0: Punkt vor dem ersten Kindknoten  $K_1$ ,  $n > 0$ : Punkt hinter  $K_n$
  - Fall „Knoten kann keine Kindknoten haben“ (*char. point*):
    - Der Index bestimmt die Lage des Punktes innerhalb des Kontextstrings des Containerknotens.
    - 0 positioniert genau vor das erste Zeichen des Knotens.

## XPointer: Punkte und Bereiche

- **Bereiche**

- Ein Bereich besteht aus zwei Punkten, aufsteigend in Dokumentenreihenfolge sortiert.
- „*collapsed range*“: Startpunkt = Endpunkt
- Stringwert eines Bereiches:
  - Fall: Zwei *character points* mit demselben Containerknoten:  
Der von den Punkten eingeschlossene String
  - Sonst  
Die Zeichen der eingeschlossenen Textknoten.

## XPointer: Neue Funktionen

- **Bereiche:**

- range-to()**

Zwei XPath-Ausdrücke zusammen mit dieser Funktion definieren einen Knotenbereich:

```
xpointer( id("ID01")/range-to(id("ID06")) )
```

- string-range()**

```
...#xpointer( string-range( //title, "XML" ) )  
...#xpointer( string-range( //title, "XML", 3, 5 ) )
```

Liefert eine Liste von Bereichen. Jeder Bereich entspricht einem <title>-Element, in dessen Text die Zeichenkette „XML“ erscheint.

Die erste optionale Zahl bestimmt den Bereich-Offset, die zweite optionale die max. Länge der gewünschten Bereiche.

## XPointer: Neue Funktionen

- **Bereiche:**

`string-range()` (weitere Beispiele)

```
string-range( //title, "XML", 2 )[5]
```

Liefert einen Bereich. Er beginnt vor dem „M“ im 5.Vorkommnis des Strings „XML“ aller Elemente <title> und endet hinter dem „L“ desselben Wortes.

```
string-range( /, "!", 1,2 )[5]
```

Liefert das fünfte Ausrufezeichen in allen Textknoten des Dokuments, zusammen mit dem folgenden Zeichen.

## XPointer: Neue Funktionen

- **Bereiche:**

`range()`

```
range( location set )  
range( //title )
```

Wandelt die übergebene Menge in Bereiche. Im Beispiel erhält man alle Bereiche der <title>-Elemente, incl. Start- und Ende-tag.

`range-inside()`

```
range-inside( location set )  
range-inside( /book/title )
```

Analog `range()`, liefert aber nur den Bereich der eingeschlossenen Knoten. Es wird auf ganze Knoten „aufgerundet“

## XPointer: Neue Funktionen

- **Punkte:**

`start-point()`

```
start-point( location set )  
start-point( string-range( //title, "XML")[1] )
```

Positioniert einen gedachten „cursor“ z.B. genau vor das erste Zeichen eines *character point*.  
Ungültig für Attribut- und Namensraumknoten.

`end-point()`

```
end-point( location set )  
end-point( string-range( //title, "XML")[1] )
```

Positioniert einen gedachten „cursor“ z.B. genau hinter das letzte Zeichen eines *character point*.  
Ungültig für Attribut- und Namensraumknoten.

## XPointer: Neue Funktionen

- **Sonstige Funktionen:**

`here()`

Liefert den Knoten, in den ein XPointer zeigt. Ausnahme:  
Zeigt der XPointer in einen Textknoten, liefert `here()` seinen Elementknoten.

`origin()`

Spezialfunktion, liefert den Ausgangsknoten einer Linkverfolgung (*traversal*) in einem XLink.  
Nur innerhalb XLink gültig! Dort näher zu besprechen.

## XPointer: *Character Escaping*

- XPointer ist ausgelegt zur Verwendung in URI Referenzen.
- In diesen Umgebungen müssen bestimmte Sonderzeichen vermieden und durch Ersatzdarstellungen repräsentiert werden (*character escaping*).
  - 1) Zeichen mit Signifikans für XPointer: `^` sowie Klammern `()()`
  - 2) Reservierte (I)URI-Zeichen:
    - `%` (immer zu ersetzen durch `%25`)
    - `[` und `]` (bei best. Software zu ersetzen durch `%5B` bzw. `%5D`)
  - 3) XML escaping: *character references*, *entity references*, ...
  - 4) Umwandlung IURI → URI (direkte Verwendung von Unicode-Zeichen ersetzen)
- Die Einzelheiten dazu bei Bedarf bitte in der XPointer-Spezifikation nachlesen, Kap. 4.1

## XPointer: Zusammenfassung

- XPointer ermöglicht präzise Verweise auf Stellen und Bereiche in XML-Dokumenten
- XPointer erschließt XPath-Ausdrücke für Links
- XPointer erweitert XPath
  - um die Objekttypen *point* und *range*
  - um Funktionen zu deren Verwendung
  - Um zwei Kurzschreibweisen (*bare names*, *child sequences*)
- XPointer wird hauptsächlich – aber nicht ausschließlich – von XLink verwendet.
  - Siehe folgendes Kapitel.

# XML Linking Language (XLink)

<http://www.w3.org/TR/xlink>

## XLink: Übersicht

- Die *XML Linking Language* – kurz XLink genannt - passt die aus HTML bekannten Link-Verfahren auf XML an:
  - Links setzen ohne fest vereinbarte Elemente?
  - Jedes XML-Element kann Ausgangspunkt werden!
  - Modernisierung: Einheitliches Vorgehen für alle Link-Typen.
- XLink präzisiert und verallgemeinert Links
  - Begriffliche Trennungen (Link, Anker, Ressource, Verweis)
  - Links in „Gegenrichtung“
  - Links ohne Anker
  - Links mit mehreren Zielen
  - Bögen – oder: Wie schließe ich Wege aus?

## XLink: HTML-Vorläufer (Beispiele)

- „Normales“ HTML-Link: Verweis
  - Durch Anklicken aktiviert
  - Neuer Inhalt ersetzt aktuelle Browser-Anzeige
  - Beispiel:

```
<a href="http://www.tagesschau.de/">
Zur Tagesschau</a>
```
- Variante: Neues Fenster
  - Durch Anklicken aktiviert
  - Neuer Inhalt erscheint in neuem Browser-Fenster
  - Beispiel:

```
<a href="http://www.tagesschau.de/"
target="_blank"> Zur Tagesschau</a>
```
- Variante: „Popup-Werbung“
  - Beim Laden der Ausgangsseite automatisch aktiviert
  - Neuer Inhalt erscheint in neuem Browser-Fenster
  - Erfordert JavaScript, Methode `window.open()`

## XLink: HTML-Vorläufer (Beispiele)

- HTML-Link zur Einbettung von Daten
  - Beim Laden der Ausgangsseite automatisch aktiviert
  - Einbettung des Inhalts (insb. Bild), ersetzt ggf. Platzhalter
  - Beispiel:

```

```
- HTML-Link zur Weiterleitung
  - Beim Laden der Ausgangsseite automatisch aktiviert
  - Neuer Inhalt ersetzt aktuelle Browser-Anzeige, evtl. zeitverzögert (im Beispiel um 5 Sekunden).
  - Beispiel (innerhalb `<head>`):

```
<meta http-equiv="refresh"
content="5; URL=http://www.mynewhome.net/">
```

## Gemeinsamkeiten der HTML-Beispiele

- Gemeinsamkeiten in der Vielfalt der HTML-Links:
  - „Was soll passieren“?
    - Aktion ist immer das Verfolgen eines Verweises
  - „Wie soll es ausgelöst werden“?
    - Die Verfolgung wird ausgelöst von einem Ereignis.
  - „Wohin mit dem Ergebnis“?
    - Das gefundene Ergebnis bewirkt eine Reaktion.
- Der XLink-Weg
  - XLink baut auf diesen Gemeinsamkeiten auf und vereinheitlicht die Syntax dazu.

## XLink: Grundregeln

- XLink stellt per **Namensraumkonvention** spezielle **globale Attribute** zur Verfügung.
- Mittels dieser Attribute kann jedes Element eines XML-Dokuments Ausgangspunkt eines Links werden.
- Beispiel für ein einfaches XLink vom Typ „Verweis“:

```
<doc xmlns:xlink="http://www.w3.org/1999/xlink">
...
<para xlink:type="simple" xlink:href=
"http://www.myhome.net/addon.xml#xpointer(...)">
Hier finden Sie weitere Informationen.
... </para>
...
</doc>
```

## XLink: Einfache Links, Attribute

Die wichtigsten XLink-Attribute für einfache Links:

- `xlink:type`
  - Bestimmt den Linktyp; in diesem Kontext fest: "simple"
- `xlink:href`
  - Spezifiziert den URI der zu erreichenden Ressource

## XLink: Einfache Links, Attribute

Die wichtigsten XLink-Attribute für einfache Links  
(Forts.):

- `xlink:show`
  - Definiert, wie die Ressource verarbeitet werden soll.
    - "new" Neues Fenster öffnen
    - "replace" Im vorhandenen Fenster anstelle des bisherigen Inhalts anzeigen
    - "embed" Im vorhandenen Fenster anzeigen und in das bisherige Material einbetten (z.B. ein Bild)
    - "other" Anzeigeverhalten bleibt der Anwendung überlassen, evtl. bestimmt durch Markup an anderer Stelle
    - "none" Anzeigeverhalten bleibt der Anwendung überlassen.

## XLink: Einfache Links, Attribute

Die wichtigsten XLink-Attribute für einfache Links (Forts.):

- `xlink:actuate`
  - Definiert, welches Ereignis die Aktion auslöst.
    - `"onRequest"` In einem grafischen Browser z.B. das „Anklicken“
    - `"onLoad"` Das Laden der Seite, die das Link trägt
    - `"other"` Auslöseverhalten bleibt der Anwendung überlassen, evtl. bestimmt durch Markup an anderer Stelle
    - `"none"` Auslöseverhalten bleibt der Anwendung überlassen

## XLink: Einfache Links, Defaults

Defaultregeln

- `xlink:href`
  - Optional bei einfachen Links!
  - Konsequenz: Ein Link kann also schon angelegt werden und damit Ereignisse auslösen (etwa in XSLT), aber erst später wirksam geschaltet werden.
- `xlink:type`
  - Muß bei einfachen Links den festen Wert `"simple"` tragen.
- `xlink:show`, `xlink:actuate`
  - Optional, Defaultverhalten bleibt dem Browser überlassen.
  - Konsequenz: Immer selbst definieren!
    - Tip dazu aus den Spezifikationen: DTD-Defaultbelegungen für Attribute nutzen!

## XLink: Einfache Links, Übung

- **Vorlesungsübung:**
  - Wandeln Sie die o.g. 5 HTML-Beispiele in XLink-Beispiele um. Erstellen Sie dazu folgende zwei Ergebnisse:
    - Ergebnis A: Tabelle mit den Spalten
      - Beschreibung des Linkverhaltens
      - Wert von xlink:show
      - Wert von xlink:actuate
    - Ergebnis B: XML-Code, der den HTML-Beispielen entspricht
- **Hinweise**
  - 5 Minuten Zeit für eigene Bearbeitung, dann Besprechung.
  - Verwenden Sie in den ersten 3 Beispielen `<a>`, dann `<img>` und schließlich `<meta>`, als wären es XML-Elemente.

## XLink: Einfache Links, Übung

Beschreibung des Links	Wert von xlink:show	Wert von xlink:actuate
Normales Link		
Anzeige in neuem Fenster		
Popup-Fenster		
Bild einbinden		
Weiterleitung		



## XLink: Erweiterte Links

*(extended links)*

## XLink: Szenarien

- **Notizen „auf“ die Online-Lektüre schreiben**
  - Verschiedene „Rollen“ der Anmerkungen:
    - Randnotizen: Zusatzangaben, Assoziationen, Bemerkungen
    - Textmarker-artige Hervorhebung, Unterstreichungen
    - Korrekturen, usw.
    - Ergänzungen per Post-It oder hineingelegter Seite
- **Was das Papier-Vorbild nicht kann:**
  - Das eBook oder Online-Dokument bleibt unverändert
  - Eigene Notizen zum Werk mit anderen teilen, fremde Notizen importieren
  - Vorhandene Notizen in die nächste Ausgabe des Werks hinüberretten (mit möglichst wenig Aufwand)
  - Echte Verweise, etwa zu anderen Dokumenten, hinzufügen.

## XLink: Szenarien, weiterführend

- **Kommerzielle Weiterentwicklungen**

- *Knowledge management*
  - Abteilungs- bzw. konzernweites Teilen von assoziativem Wissen durch passende Organisation einer gemeinsamen Anmerkungs-Datenbank zum Dokumentenbestand
- Beispiel „Technische Handbücher“:
  - Sammlung von Anwenderkommentaren könnte Schwachstellen rasch erkennbar machen.
  - Vernetzung der Information, etwa mit Hintergrundartikeln oder Verweisen in Stamm- oder Bestandsdatenbanken
- Neue Dienstleistungen
  - Entkopplung der Autorentätigkeit von der Vernetzung der Werke (alter wie neuer), Einkauf von Vernetzungsleistungen
  - „Guided tours“

## XLink: Szenarien, weiterführend

- **Anmerkungen**

- Beispiele zu den genannten Szenarien existieren längst.
- Sie haben aber den Charakter von Insellösungen, da sie auf proprietären Techniken beruhen.
- Die XLink-Spezifikationen stellen den Versuch dar, derartige Konzepte in den *mainstream* zu integrieren.
- Erst dadurch wird ein hinreichend großer Markt geschaffen.
- Vision „*topic maps*“: XLink kann ein entscheidender Beitrag werden zur semantischen Vernetzung des Weltwissens.

- **Status**

- XLink ist von der praktischen Umsetzung noch weit entfernt.
- Anwendungen sind erst nach weiterer Verbreitung von nativen XML-Dokumenten zu erwarten.
- Einfache XLinks sollten jedoch schon heute verwendet werden, z.B. um sie per XSLT in ihre HTML-Pendants zu konvertieren.

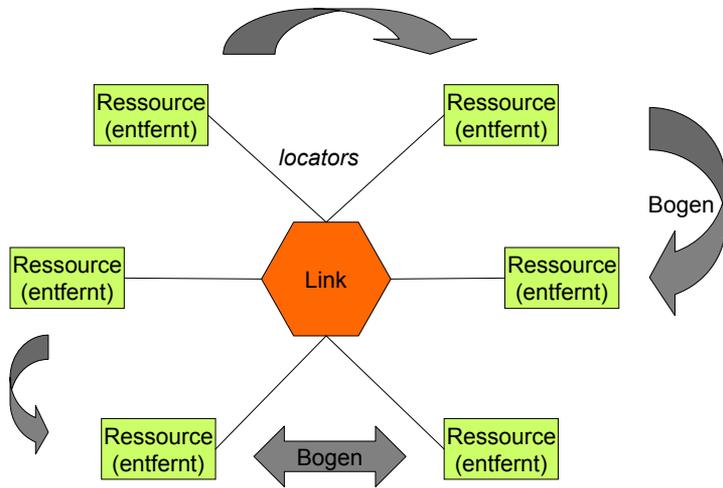
## XLink: Szenarien, Weg dorthin

- Was benötigen wir, um derartige Szenarien technisch zu realisieren?
  - Links, die in „Rückwärtsrichtung“ funktionieren (etwa vom Buch zur Anmerkung).
  - Links, die unabhängig von ihren Ressourcen existieren (sonst kann man sie nicht separat handeln oder tauschen).
  - Linksammlungen (die man einem Werk beilegen kann oder mit dem Browser separat einbinden kann).
  - Anwendungen wie Browser, die Verknüpfungspunkte solcher Linksammlungen im jeweiligen Dokument anzeigen.
  - Anwendungen, die ggf. mehrere Link-Alternativen zur Auswahl anbieten, und die rollenabhängig auf Links reagieren („Nun bitte ‚Textmarker an‘, ‚Hyperlinks aus‘, ‚Randnotizen als *hover*-Texte über *hot spots*‘), und und und.

## XLink: Der Link-Begriff

- Was bedeutet „Link“ genau?
  - Verkettung, Verbindung, Verknüpfung
- Was wird verknüpft?
  - 2 oder mehr (!) Ressourcen
- Wie beschreibt man Ressourcen?
  - Durch Namen und Rollen sowie Etiketten (*label*)
- Und die Wege dorthin?
  - Durch Angaben, wie man sie findet, i.a. als *URI* bzw. *URI reference*.
  - Mittels eigener Objekte, sog. *locators*. Diese erhalten Attribute analog zu denen der Ressourcen.
- Bewirkt die Verknüpfung von Ressourcen schon etwas?
  - Nein, das bekannte Verfolgen von Links (*traversal*, wörtlich „queren“, „passieren“) benötigt i.a. noch „Bögen“ (*arcs*).
- Was sind denn Bögen?
  - Mit Bögen beschreibt man die zugelassenen „Passagen“ zwischen Ressourcen.
  - Mit Bögen unterschiedlicher Rollen können Links je nach Kontext eingeblendet werden.
- Warum so kompliziert???
  - Siehe Szenarien!

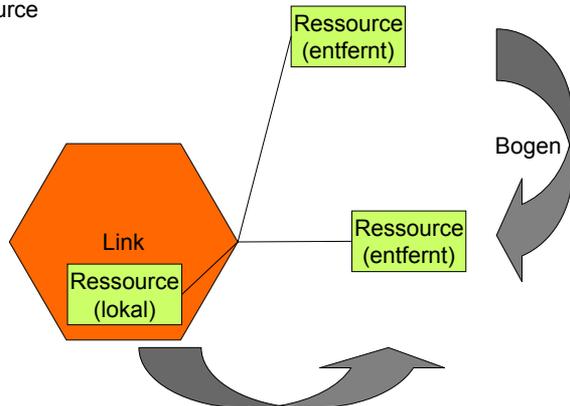
# XLink: Der Link-Begriff



# XLink: Der Link-Begriff

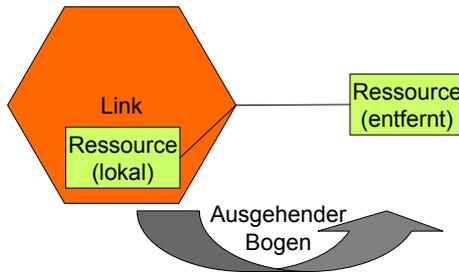
## Variante eines Links:

Mit lokaler Ressource



# XLink: Der Link-Begriff

## Das einfache Link als Spezialfall des erweiterten Links



- Die lokale Ressource ist Ausgangspunkt des Bogens
- Sie trägt entweder selbst das Link oder ist ein Kind-element des Linkelements

# XLink: Das Attribut `type`

- Zulässige Werte von `type`
  - `simple` siehe früheren Abschnitt
  - `resource` Definiert ein Element als Ressource
  - `title` Weist ein Element als Titel aus, den ein XLink-fähiger Browser zu Anzeigezwecke nutzen könnte
  - `locator` Weist ein Element als Träger des href-Attributs aus Kann noch weitere Informationen aufnehmen
  - `arc` Leitet ein Element ein, dessen Attribute einen Bogen („passierbaren Weg“) definieren
  - `extended` Kennzeichnet das Basiselement aller erweiterten Links

## XLink: Eltern-Kind-Beziehungen

Elterntyp	Kind-Typen
simple	(keine)
extended	locator, arc, resource, title
locator	title
arc	title
title	(keine)
resource	(keine)

## XLink: Attributvorkommen

	simple	extended	locator	arc	resource	title
type	R	R	R	R	R	R
href	O		R			
role	O	O	O		O	
arcrole	O			O		
title	O	O	O	O	O	
show	O			O		
actuate	O			O		
label			O		O	
from				O		
to				O		

## XLink: Attributgruppierung

- Elementtyp-Attribut
  - `type`
- Locator-Attribut
  - `locator`
- Semantische Attribute
  - `role`
  - `arcrole`
  - `title`
- Attribute zur Verhaltenssteuerung
  - `show`
  - `actuate`
- Attribute zur Linkverfolgung
  - `label`
  - `from`
  - `to`

## XLink: Beispiel

- Vorbemerkungen
  - Der Umgang mit *extended links* soll hier nicht erschöpfend behandelt werden.
  - Statt dessen wird ein Code-Beispiel vorgestellt und diskutiert.
  - Die erweiterten Möglichkeiten von *extended links* sowie das allgemeinere Konzept werden damit bereits sichtbar.
  - Gewähltes Beispiel: Ein einfaches Link, als *extended link* nachempfunden, frei nach den XLink-Spezifikationen.

## XLink: Beispiel

- Anwendung analog <a>:

```
..., und <studentlink  
  xlink:href="students/patjones62.xml">Pat  
  Jones</studentlink> nähert sich nun dem Diplom.
```

- Funktioniert z.B. mit folgenden DTD-Einträgen:

```
<!ELEMENT studentlink ANY>  
<!ATTLIST studentlink  
  xlink:type (simple) #FIXED "simple"  
  xlink:href CDATA #IMPLIED  
  xlink:role NMTOKEN #FIXED  
    http://www.example.com/linkprops/student  
  xlink:arcrole CDATA #IMPLIED  
  xlink:title CDATA #IMPLIED  
  xlink:show (new|replace|embed|other|none) #IMPLIED  
  xlink:actuate (onLoad|onRequest|other|none) #IMPLIED>
```

## XLink: Beispiel

- Und nun das Link in „extended-Version“, Teil 1:

```
<studentlink xlink:type="extended">  
  <resource  
    xlink:type="resource"  
    xlink:label="local">  
    Pat Jones</resource>  
  <locator  
    xlink:type="locator"  
    xlink:href="students/patjones62.xml"  
    xlink:label="remote"  
    xlink:role="..."  
    xlink:title="..."/>
```

## XLink: Beispiel

- „extended-Version“, Teil 2:

```
<go
  xlink:type="arc"
  xlink:from="local"
  xlink:to="remote"
  xlink:arcrole="..."
  xlink:show="..."
  xlink:actuate="..." />
</studentlink>
```

- Kind-Elemente von <studentlink>:
  - resource Trägt den Nutztext; Lokales Ende
  - locator Beschreibt den Weg zum entfernten Ende
  - go Beschreibt einen ausgehenden „Bogen“ von Label „local“ zu Label „remote“

## XLink: *Linkbases*

- Ein Nachtrag zum Attribut `xlink:role`
  - `role` und `arcrole` nehmen nur URI als Attributwerte auf.
  - Es sind abstrakte *URI*, ähnlich wie *namespace URI*, d.h. sie zeigen auf keine Ressourcen, sondern dienen global eindeutigen Kennzeichnungen.
  - Wenn Sie eigene Rollen zu benennen haben, so leiten Sie Ihre Rollen-*URI* z.B. von Ihrem Domainnamen ab.
- Standard-URI (Beispiel *linkbase*):
  - Sie können einem Browser mitteilen, wo er eine Linksammlung (*linkbase*) zum aktuellen Dokument findet, indem Sie dem Link das Attribut `arcrole` mit Wert <http://www.w3.org/1999/xlink/properties/linkbase> zuweisen. Siehe auch die Szenarien für die Frage „warum“.
  - Mit weiteren öffentlichen Rollen-URI ist zu rechnen.

## XLink: Diskussion

- Alles hat seinen Preis
  - Die Verallgemeinerung der Linkbildung erzeugt gewaltigen *overhead* und neue Komplexität.
  - Die dadurch geschaffenen Möglichkeiten erfordern geeignete neue Werkzeuge
    - Anwendungen wie Browser, die XLink unterstützen
    - Entwicklungs- und Autorenwerkzeuge mit grafischer Oberfläche, die den (XLink-konformen XML-) Code vor den Anwendern verbergen
  - Erst eine geeignete Infrastruktur wird die neuen Möglichkeiten erschließen
    - Direkter Zugriff auf XML-Dokumente, öffentliche Linksammlungen zu gängigen Dokumenten, ...

## XBase: XML Base

Ein nützlicher kleiner Standard

<http://www.w3.org/TR/xbase>

# XBase

- Zweck von XML Base:
  - Umdefinieren des Bezugspunktes von relativen URLs.
- Szenario:
  - Sie bauen eine Website aus XML-Dokumenten auf und verwenden dabei zahlreiche relative URLs wie `"../pics/pic1.gif"`, `"chapter03.xml"` etc.
  - Sie kopieren die Daten an einen anderen Standort.
  - Fall 1: Die relativen URLs sollen nur innerhalb des neuen Standorts funktionieren. Kein Handlungsbedarf!
  - Fall 2: Ihr XML-Hauptdokument, das zahlreiche relative URLs enthält, soll überall auf die Originale an Ihrem alten Standort zeigen!
  - Entweder Sie editieren alle URLs um - oder nur eine Zeile!

# XBase

- XML Base: So geht's
  - Fügen Sie das globale Attribut `xml:base` ein in das hierarchisch niedrigste Element, unterhalb dessen alle Ihre umzudefinierenden relativen URLs stehen. Im einfachsten Fall ist dies das Dokumenten-Element.
- Beispiel:

```
<mydoc xmlns:xlink=...
  xml:base="http://www.myoldhome.org/">
...
  <citation xlink:type="simple"
    xlink:href="litref03.xml">
...
</mydoc>
```