



***XML Base (XBase),  
XML Pointer Language (XPointer),  
XML Include (XInclude),  
XML Linking Language (XLink),***

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

<http://www.w3.org/TR/xptr-framework>

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

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



# XML Base

Ein nützlicher kleiner Standard

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

REC 27.06.2001



- **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 "`../picts/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 dank XML Base nur eine Zeile!



- XML Base: So geht's
  - Fügen Sie das globale Attribut `xml:base` ein in das hierarchisch niedrigste Element, unterhalb dessen alle Ihre neu zu definierenden 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>
```

Expandiert nun zu:

<http://www.myoldhome.org/litref03.xml>



# ***XML Pointer Language***

Übersicht, Ursprünge  
Konzepte  
Erweiterungen



- **Was ist XPointer?**

- Die Spezifikation zur XML-konformen Formulierung von Verweisen auf Dokumente, Dokumentteile und –bereiche.
- Ein Hilfsstandard,
  - der XLink (s.u.) zuarbeitet und
  - auf XPath aufbaut
- Eine Verallgemeinerung der *URI* und *URI references* aus HTML.
- Eine Serie von W3C-Empfehlungen (Framework: 25.3.2003)

- **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 gestatten in der Tat bereits das Adressieren von XML-Dokumentteilen, müssen aber noch „verpackt“ werden, um gültige Links zu ergeben.
- XPointer erweitert XPath um Referenzierungskonzepte:
  - Ortsangaben
  - Bereichsangaben

- **Erinnerung: URI-Referenzen bestehen aus**

- dem **URI-Teil**
- dem Trennzeichen ( # ) sowie
- dem *fragment identifier*
- Beispiel: `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>`
  - XHTML 1.1-Version:  
`<h2><a id="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“)
- **Nachteile von HTML-Ankern**
  - Verweise nur zu vorab eingerichteten Ankern / Zielen
  - Konzeptionell begrenzt:
    - Keine Bereiche adressierbar
    - Nicht zeichengenau
    - Unflexibel



- **Neue Möglichkeiten durch XPointer**

- Verweise auf präzise Stellen in XML-Dokumenten ganz ohne „Anker“, Verweise auf Bereiche
- 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.

- Umsetzung mit neuen XPath-Funktionen und –datentypen



- **URI mit XPointer**

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

- ...#**xpointer**( *XPath\_expression* )

- (sowie einige andere Fälle („schemes“))

- Formale Grundlage:

- <http://www.w3.org/TR/xptr-framework/> REC, 25.3.2003

- Beispiel:

- <http://www.myorg.com/>

- [foo.xml#xpointer\(//table-of-content\[1\]\)](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“), z.B. um Ersatzorte anzugeben, falls die primäre Referenz nicht mehr besteht.



- **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.mysample.com/ns/a1)  
  xpointer (//pfx:toc)
```

- Die Namensraumverwaltung im Dokument selbst ist davon unabhängig:

```
<my:doc xmlns:my="http://www.mysample.com/ns/a1">  
  <my:toc>Table of Content  
  <my:item>...</my:item></my:toc>... </my:doc>
```

- Formale Grundlage:

```
http://www.w3.org/TR/xptr-xmlns/ REC, 25.3.2003
```



- **Datenmodell von XPointer**

- XPointer's Datenmodell kommt dem von XML Infoset näher als dem von XPath, denn einzelne Zeichen sind adressierbar.
- Ähnlich wie XSLT gestattet auch XPointer die Anwendung auf Dokumentfragmente, wie sie z.B. in externen *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 eine **Kurzschreibweise** ein.
- **Neue Funktionen** unterstützen das *location*-Konzept



- **Kurzschreibweise** (vormals „*barenames*“):

- Für die Verwendung von `id()` gilt:

Statt `...#xpointer( id("ID12345") )`

kann man auch nur die ID (ohne " ") schreiben:

`...#ID12345`

- **Bemerkungen**

- **SVG** verwendet diese Technik (vermittels XLink), z.B. in:

```
<defs>
```

```
  <symbol id="KK" stroke-width="4" >
```

```
    <line x1="0" y1="0" x2="0" y2="50" />
```

```
    <line x1="0" y1="0" x2="50" y2="0" stroke="blue"/>
```

```
  </symbol>
```

```
</defs>
```

```
<use xlink:href="#KK" transform="scale(4 3)"/>
```

- **Vorsicht:** Formal identisch zur alten *fragment identifier*-Notation, aber technisch völlig verschieden!



# XPointer: element()



- Das Konstrukt `element()` (vorm. „*child sequences*“):
  - Elemente im Dokumentenbaum können auch durch eine Art Index angesprochen werden:  
`...#element (ID12345/1/2)`
  - Bedeutung: Das zweite Unterelement des ersten Unterelements des Elements mit der ID „ID12345“
- Bemerkungen
  - Lässt sich alternativ und in Kombination zu `xpointer()` verwenden
  - Index zählt ab 1 (wie bei XPath)
- Formale Grundlage:
  - <http://www.w3.org/TR/xptr-element/> REC, 25.3.2003



# XPointer: xpointer()



- Das Konstrukt `xpointer()`:
  - Elemente im Dokumentenbaum werden in der Regel über einen XPath-Ausdruck als Argument von `xpointer()` adressiert:  
`...#xpointer(//section[3])`
  - Bedeutung: Das dritte Unterelement „section“ der Zielressource (ganzes XML-Dokument wie auch externes Entity zulässig)
- Formale Grundlage:
  - <http://www.w3.org/TR/xptr-xpointer/>  
Noch immer Entwurfsstatus, zuletzt vom 19.12.2002
- Stellt neue XPath-Funktionen bereit:
  - `range-to()`, `string-range()`
  - `covering-range()`, `range-inside()`, `start-point()`, `end-point()`
  - `here()`, `origin()`



- **Der Punktbegriff**

- Um zeichengenau adressieren zu können, definiert XPointer zwei neue Knotentypen für Positionsangaben, *point* und *range*.
- XPath-Ausdrücke lassen sich in XPointer auch auf diese Knotentypen 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.



- **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.**



- 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.



- **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.



- **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.



- **Bereiche:**

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



- **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 Signifikanz 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 I(U)RI → URI  
(direkte Verwendung von Unicode-Zeichen ersetzen)
- Die Einzelheiten dazu bei Bedarf bitte in der XPointer-Framework Spezifikation nachlesen, Kap. 4



# XPointer: Zusammenfassung

---



- XPointer ermöglicht präzise Verweise auf beliebige Stellen und Bereiche in XML-Dokumenten, auch ohne deren vorherige Markierung.
- 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 Inclusions (XInclude)***

<http://www.w3.org/TR/xinclude>  
V 1.0, REC (2. Auflage), 15.11.2006



# XInclude



- XInclude beschreibt eine Syntax und ein Verarbeitungsmodell zur Vereinigung mehrerer XML-Infosets in ein Komposit.
- Die Art der Vereinigungsbildung wird mit Elementen, Attributen und URI-Referenzen gesteuert.
- XInclude stellt dazu Elemente eines reservierten Namensraums bereit:  
<http://www.w3.org/2001/XInclude>

- **Beispiel:**

```
<?xml version='1.0'?>
<document xmlns:xi="http://www.w3.org/2001/XInclude">
  <p>120 Mz is adequate for an average home user.</p>
  <xi:include href="disclaimer.xml"/>
</document>
```

Wirkung: Inhalt von „disclaimer.xml“  
wird nach „p“ eingefügt, vgl. **Demo**.

```
<?xml version='1.0'?>
<disclaimer>
  <p>The opinions represented herein represent those of the
    individual and should not be interpreted as official
    policy endorsed by this organization.
  </p>
</disclaimer>
```



# XInclude vs. *external entity*-Referenzen



- **Alternativ-Beispiel, mit *ext. entity*-Ansatz:**

```
<?xml version='1.0'?>
<!DOCTYPE document [
<!ENTITY disclaimer SYSTEM "disclaimer.xml" >
]>
<document>
  <p>120 Mz is adequate for an average home user.</p>
  &disclaimer;
</document>
```

- **Die beiden Wege sind NICHT äquivalent:**
  - Ext. Entities verändern das Originaldokument (und somit auch sein Parse-Verhalten, seine Validierung und sein Infoset)
  - XInclude lässt das/die ursprüngliche(n) Infoset(s) unverändert und erzeugt ein NEUES Komposit-Infoset – jenseits der Parse- und Validierungsschritte.



- Einbinden von CDATA-Texten

```
<?xml version='1.0'?>
<document xmlns:xi="http://www.w3.org/2001/XInclude">
  <p>This document has been accessed
  <xi:include href="count.txt" parse="text"/>
  times.</p>
</document>
```

- Spezialfall: Wörtliches Einbinden von Quelltexten

```
<?xml version='1.0'?>
<document xmlns:xi="http://www.w3.org/2001/XInclude">
  <p>The following is the source of the "data.xml" resource:
  <xi:include href="data.xml" parse="text"/>
  times.</p>
</document>
```

Demo! - Beachten Sie die  
Behandlung von Markup-Zeichen



- **Das Attribut „xpointer“**

- XInclude kann auch mittels XPointer bezeichnete Bereiche aus XML-Dokumenten auszugsweise adressieren!

- **Beispiel**

```
<?xml version='1.0'?>
<price-quote xmlns:xi="http://www.w3.org/2001/XInclude">
  <prepared-for>Joe Smith</prepared-for>
  <good-through>20040930</good-through>
  <xi:include href="price-list.xml" xpointer="w002-description"/>
  <volume>40</volume>
  <xi:include href="price-list.xml"
    xpointer="element (w002-prices/2)"/>
</price-quote>
```

Einzelheiten:  
Siehe **Demo!**



# XInclude und XPointer



- Fortgeschrittene XPointer-Anwendung
  - Zeichengenaue Bereichsdefinition

- **Beispiel**

```
<?xml version='1.0'?>
<document>
  <p>The relevant excerpt is:</p>
  <quotation>
    <include xmlns="http://www.w3.org/2001/XInclude"
      href="source.xml" xpointer=
        "xpointer(string-range(chapter/p[1], 'Sentence 2')/
          range-to(string-range(/chapter/p[2]/i, '3.', 1, 2)))"
    />
  </quotation>
</document>
```

Einzelheiten:  
Siehe **Demo!**



# XInclude: *Fallbacks*



- **Ersatzdaten, falls Quelle nicht verfügbar**
  - Angabe von Ersatzquellen oder –daten mit Element „fallback“
  - Auch kaskadierbar, siehe Beispiel:

- **Beispiel**

```
<?xml version='1.0'?>
<div>
  <xi:include href="example.txt" parse="text"
             xmlns:xi="http://www.w3.org/2001/XInclude">
    <xi:fallback> _____ Äußeres fallback
      <xi:include href="fallback-example.txt" parse="text">
        <xi:fallback> _____ Fallback zum fallback
          <a href="mailto:bob@example.org">Report error</a>
        </xi:fallback>
      </xi:include>
    </xi:fallback>
  </xi:include>
</div>
```

Einzelheiten:  
Siehe **Demo!**



# ***XML Linking Language (XLink)***

<http://www.w3.org/TR/xlink>  
REC, 27.06.2001



- 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()`



- 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 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>
```



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



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

- **xlink:show**
  - Definiert, wie die Ressource verarbeitet werden soll.
  - Zulässige Attributwerte:

"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.



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

- **xlink:actuate**

- Definiert, welches Ereignis die Aktion auslöst.
- Zulässige Attributwerte:

"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



## Defaultregeln

- **xlink:href**
  - Bei einfachen Links optional!
  - 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.  
Tipp dazu aus den Spezifikationen:  
DTD-Defaultbelegungen für Attribute nutzen!



- **Vorlesungsübung:**

- Wandeln Sie die o.g. 5 HTML-Beispiele in XLink-Beispiele um.

Erstellen Sie dazu folgende zwei Ergebnisse:

- Ergebnis A: Tabelle mit folgenden 3 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.



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



## 1. Normales Link

```
<a xlink:href="http://www.tagesschau.de/"  
    xlink:show="replace" xlink:actuate="onRequest">  
    Zur Tagesschau</a>
```

## 2. Neues Fenster

```
<a xlink:href="http://www.tagesschau.de/"  
    xlink:show="new" xlink:actuate="onRequest">  
    Zur Tagesschau</a>
```

## 3. Popup-Werbung

```
<a xlink:href="http://www.werbung.xy/"  
    xlink:show="new" xlink:actuate="onLoad">
```

## 4. Bild einbinden

```
<img xlink:href="mypic.gif" alt="Hier ..."  
    xlink:show="embed" xlink:actuate="onLoad">
```

## 5. Weiterleitung

```
<meta xlink:href="http://www.mynewhome.net/"  
    xlink:show="replace" xlink:actuate="onLoad">
```



- Demos:
  - XLink-Unterstützung durch die heutigen Browser  
Was funktioniert bereits mit den heutigen Browsern?  
Unterschiede: Mozilla Firefox vs. IE
  - Quelltext: Wie geht's genau?  
index.xml & teil1.xml  
linksammlung.xml, linkstore.xml
  - Gestaltungsmöglichkeiten mit CSS und XSLT
- Fazit:
  - Reine XML-Lösungen zu (einfachen) XLinks sind heute schon praktikabel mit Browsern der „XML-Generation“.