XML Path Language (XPath)

23.05.2006

Gholaman, Ali
Bui, Binh
Rodionov, Dmytro
Büscher,Miguel
Colak,Sedat
Kebiri, Yassine

Entstehung

- Wurde im November 1999 verabschiedet.
- Wurde von dem W3C in einer Zusammenarbeit zweier Arbeitsgruppen (Arbeitsgruppe für Formatierung von Dokumenten und Linking-Spezifikation) entwickelt.
- Aktuelle Version 2.0
- Gängige Version 1.0

Beziehung

Folgende W3C-Standards bauen auf Xpath auf:

- XSLT
- XML Schema
- XPointer
- XQuery (basiert auf XPath 2.0)

Definition

- Xpath ist eine Adressierungssprache, die ein Dokument als Baum bzw. Teilbaum oder Ast auffasst.
- In diesem Knotenbaum kann man mit Hilfe einer Adressierungssyntax „navigieren“.
- Die Steuerung erfolgt durch Lokalisierungsausdrücke, die ähnlich wie Dateipfade aufgebaut sind.

Datentypen

- XPath definiert vier Grundtypen:
  1. boolean (*true* oder *false*),
  2. number (Gleitkommazahl),
  3. string (Zeichenkette),
  4. node-set (***ungeordnete*** Knotenmenge).
- Die Variablendeklaration muss in der jeweilig benutzten Umgebung stattfinden, da Xpath selbst keine Variablen deklarieren kann.

Datenmodell

- <Autor>
  <Name>Mustermann</Name>
  <Vorname>Peter</Vorname>
  </Autor>
- Xpath hat ein höheres Abstraktionsniveau, als eine einfache Wortschatzanalyse.
- Die Grundstruktur eines XML-Dokuments ist aus Sicht von XPath baumartig.

XPath kennt sieben verschiedene Knotenarten:

1. Wurzelknoten (root nodes)
2. Elementknoten (element nodes)
3. Textknoten (text nodes)
4. Attributknoten (attribute nodes)
5. Namensraumknoten (namespace nodes)
6. Verarbeitungsanweisungs-Knoten (processing instruction nodes)
7. Kommentarknoten (comment nodes)

- **Wurzelknoten:** Dies ist die Wurzel eines XML-Dokuments. XPath-Knoten der das gesamte Dokument enthält. Anders als alle Elemente (und Attribute) verfügt er über keinen Namensraumbezeichner. In Xpath 2.0 zu Dokumentknoten umbenannt.

- **Elementknoten:** Für jedes Element des Dokuments existiert ein Elementknoten, der wieder weitere Elementknoten als Kinder haben kann als auch Kommentare, Verarbeitungsanweisungen und Text. Ein Element namens a, bestehend aus dem Anfangs- und End-Tag <a>.... Untergeordnete Elemente (Kindelemente) sind ebenfalls „element nodes“ - z.B. ein Element namens b, bestehend aus Anfangs- und End-Tag ..., welches innerhalb von <a>... vorkommen kann.

- **Textknoten:** Enthalten keine weiteren Knoten, entsprechen Blättern eines Baumes (vgl. DAP2).

- **Attributknoten:** Zu einem Elementknoten gehört eine optionale Menge an Attributen, die Xpath anstatt Kinderelementen des Elementknotens, als Menge von zugeordneten Knoten interpretiert. Attributknoten ist definitionsgemäß kein Kind seines Elterknotens. Die einzelnen Attributknoten enthalten Zeichenketten. //Ein Attribut b="irgendwas", das im Anfangs-Tag eines Elements a vorkommt und dann in der Form <a b="irgendwas"> notiert wird, gilt als „attribute node“ des Elements a. Der Attributknoten besteht aus dem Attributnamen b und dem zugewiesenen Wert irgendwas.

- **Namensraumknoten:** Wie bei attribute nodes sind Namensraumknoten nicht Kinder des sie enthaltenen Elementknotens, das aber als ihr Vorfahr gilt.
- **Verarbeitungsanweisungsknoten:** Wie bei Namensraumknoten ist das Ziel der Verarbeitungsanweisung der lokale Name, der Namensraumbezeichner ist Null.
- **Kommentarknoten:** Kinderlos, ohne Namensraumbezeichner; Wert entspricht der durch <!-- und --> eingeschlossenen Zeichenkette.

Jeder Xpath-Ausdruck besteht aus:

1. Achse

Beispiel.:

/Referat/Inhalt/Kapitel/attribute

Jeder Xpath-Ausdruck besteht aus:

1. Achse
2. Knotentest

Beispiel.:

/Referat/Inhalt/Kapitel/attribute :: */parent

Jeder Xpath-Ausdruck besteht aus:

1. Achse
2. Knotentest
3. Optionalen Prädikaten

Beispiel.:

/Referat/Inhalt/Kapitel/attribute :: */parent
:: node()

Absolute Pfade

- Absolute Pfade gehen von der Dokumentwurzel aus und beginnen mit einem „/“.
- Beispiel.: /Referat/Kopf
- In dem Beispiel werden zwei Location-Steps verwendet. Man wandert von dem Dokumentwurzel zum Knoten „Referat“ und vom Knoten „Referat“ zum Knoten „Kopf“.

Relative Pfade

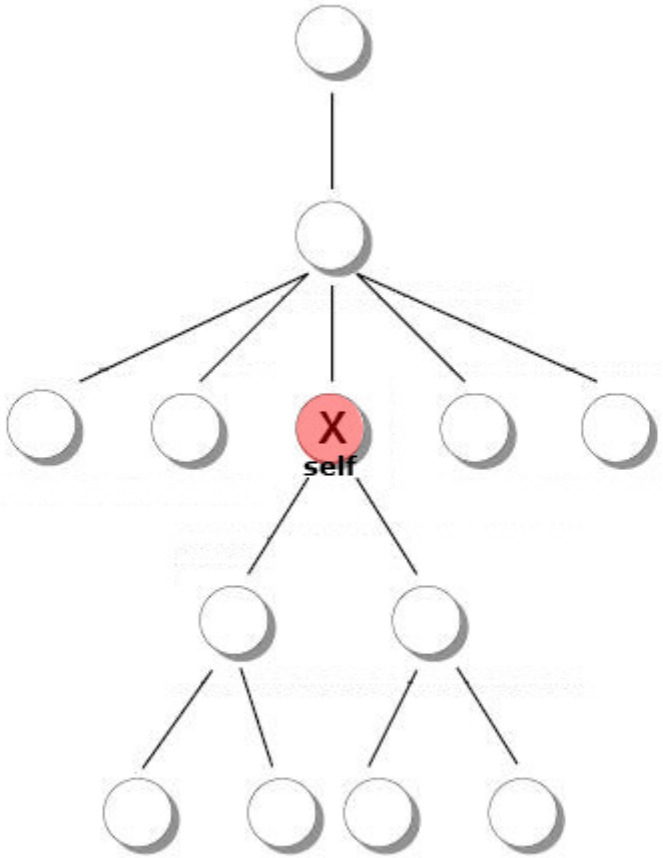
- Die Pfade gehen vom aktuellen Knoten aus, welche auch Kontextknoten genannt werden. Sie beginnen ohne führenden „/“.
- Beispiel.: Autor/Name
- Der Pfad fängt in diesem Fall dort an wo das vorherige Beispiel endet (/Kopf).

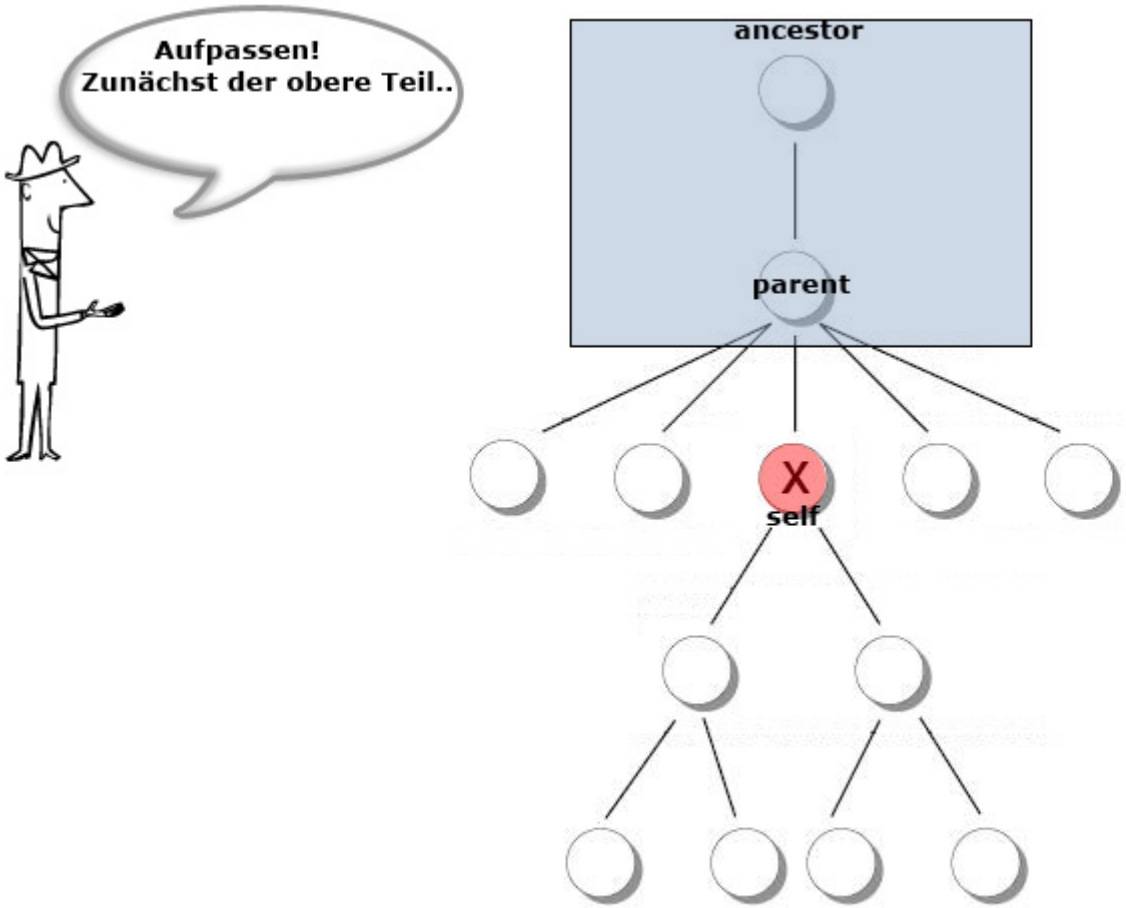
Achse-Knotentest

- Achse wird durch zwei Doppelpunkte von dem angegebenen Knoten getrennt (objektorientierte Programmierung)
- Achse::Knotentest[Prädikat]

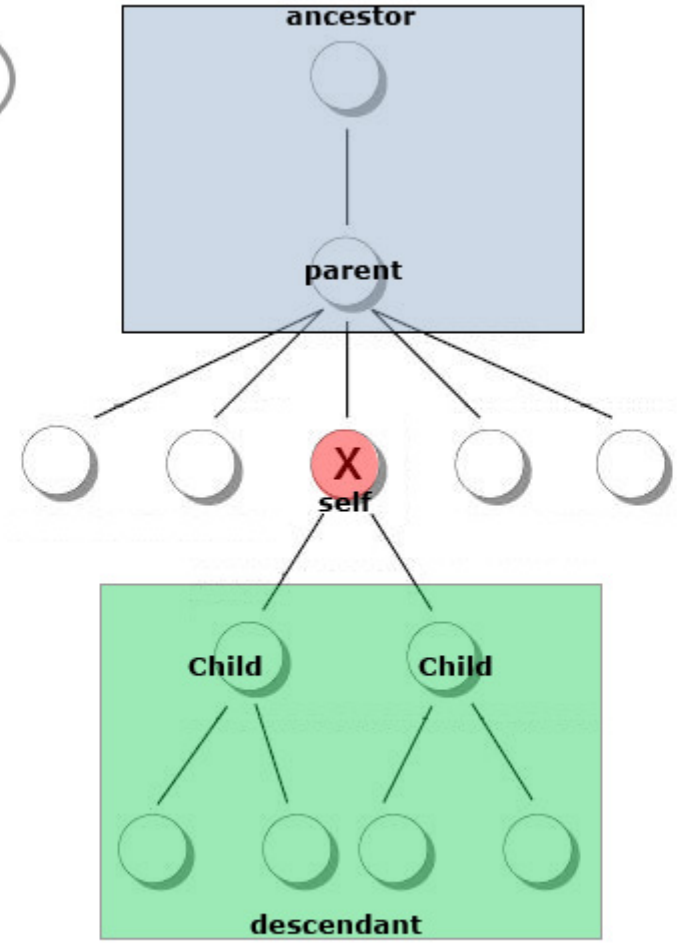
Achsen-Typen

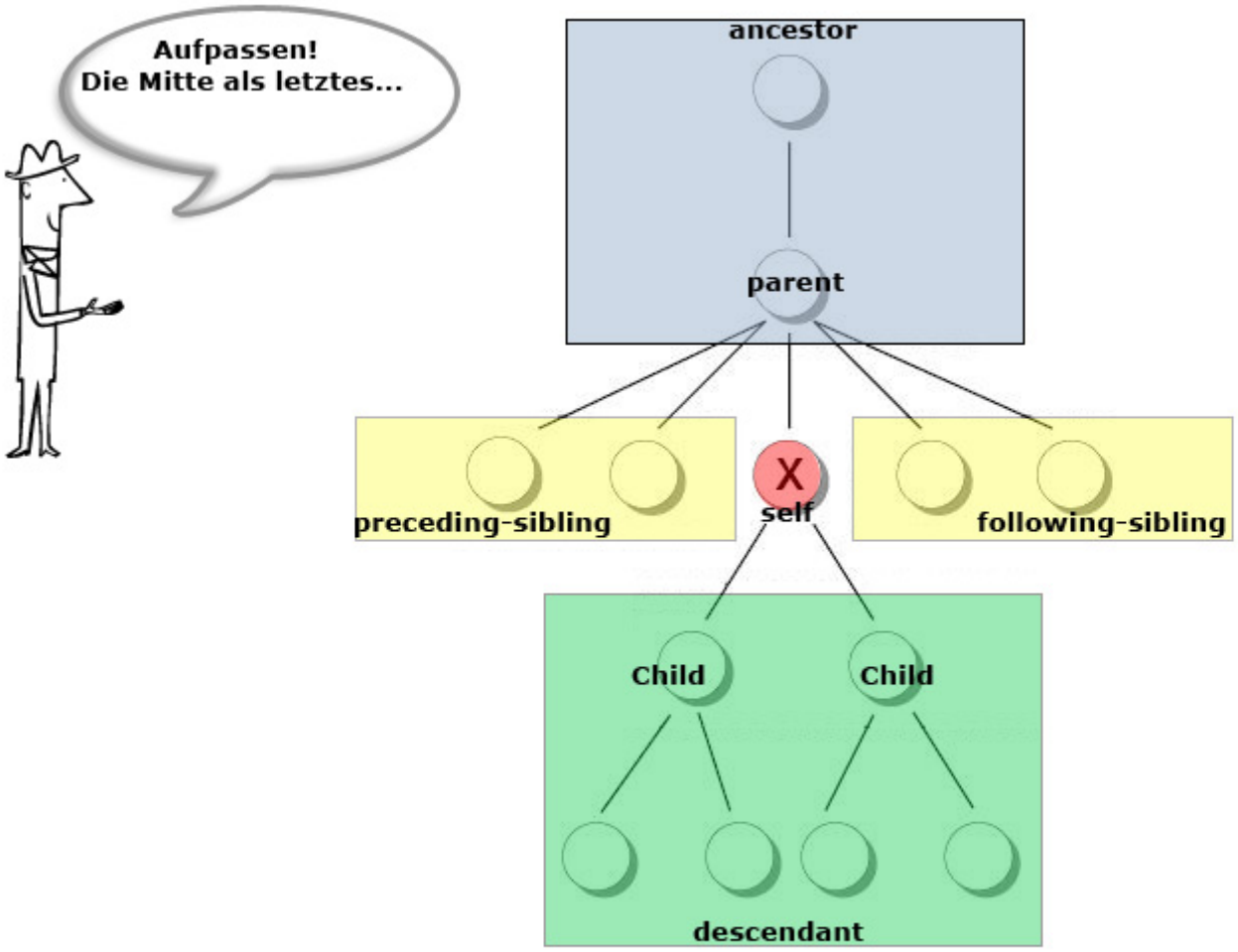
Achse	Bedeutung
child	Kindelemente
parent	Direkter Vorfahr
descendant	Nachfahren
ancestor	Vorfahren
following	Alle nach dem gegenwärtigen im Dokument vorhandenen Knoten
preceding	Alle vor dem gegenwärtigen im Dokument vorhandenen Knoten
following-sibling	Geschwister des gegenwärtigen Knotens, die noch folgen
preceding-sibling	Geschwister, die vor dem gegenwärtigen Knoten angesiedelt sind
attribute	Die Attribute eines Knotens (nur bei Elementen)
namespace	Namensraumknoten des gegenwärtigen Knotens (nur bei Elementen)
self	Der gegenwärtige Knoten
descendant-or-self	Wie descendant, plus der gegenwärtige Knoten
ancestor-or-self	Wie ancestor, plus der gegenwärtige Knoten

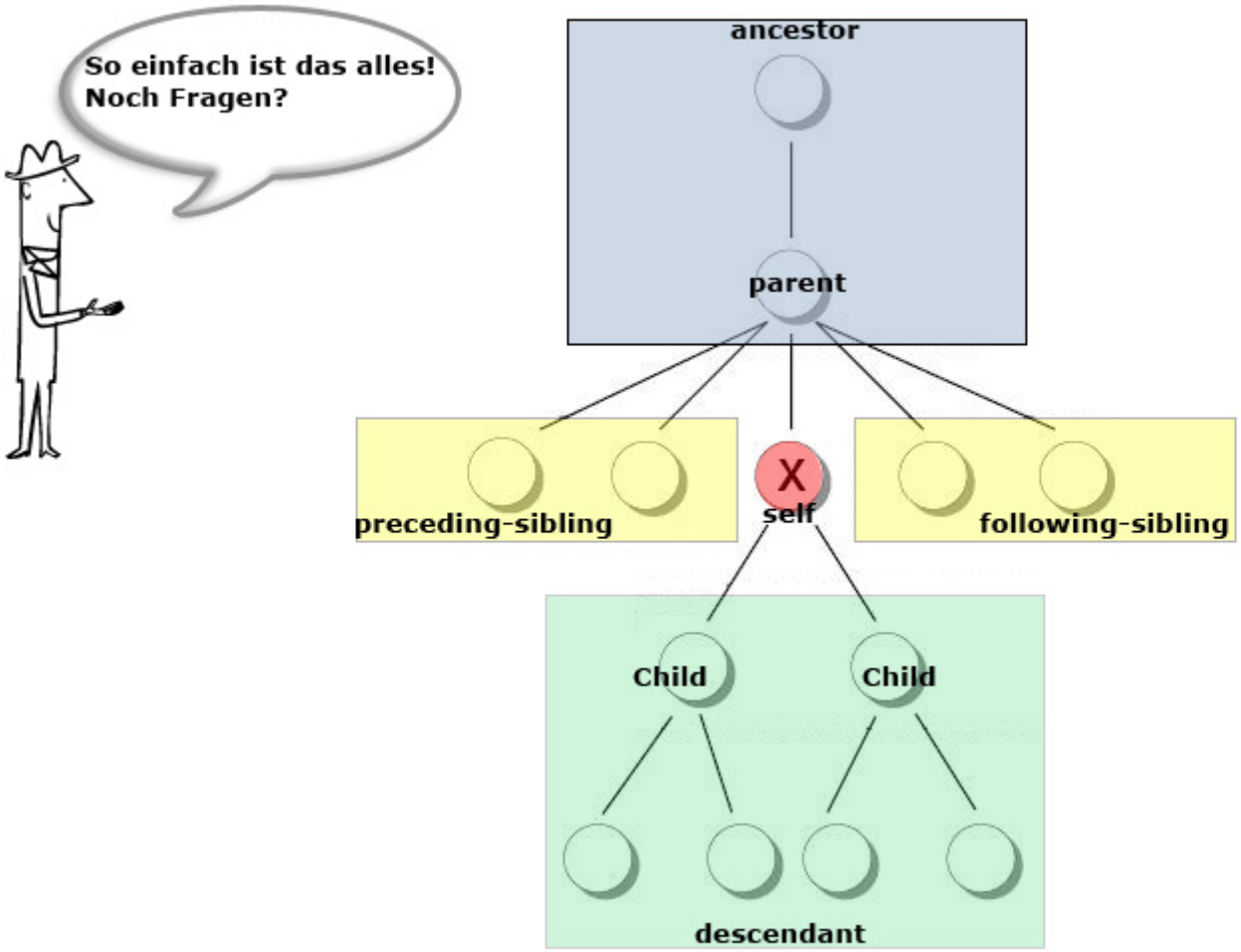

X
self


Aufpassen!
Zunächst der obere Teil..
ancestor
parent
X
self


Aufpassen!
Und nun der Untere...


ancestor
parent
X
self
Child
Child
descendant


Aufpassen!
Die Mitte als letztes...
ancestor
parent
preceding-sibling
X
self
following-sibling
Child
Child
descendant


So einfach ist das alles!
Noch Fragen?
ancestor
parent
preceding-sibling
X
self
following-sibling
Child
Child
descendant

Prädikaten-Operatoren

- Um Xpath-Ausdrücke komplexer gestalten zu wollen, braucht man bestimmte Objekte, die einen booleschen Wert ergeben. (DAP1, DAP2)
- Or, and, =, !=, <, >, <=, >=, +, -, div, mod, *
- **WICHTIG**: Vor * dürfen folgende Zeichen nicht stehen: @, ::, (, [, ,
- Das Minuszeichen ist der einzige Operator, der unär also als Vorzeichen benutzt werden kann.

Prädikaten-Ausdrücke

Ausdruck	Zugriff auf
/	Ursprung des Dokumentbaums, nicht das Top-Level-Element des Dokuments
child::*	Alle Kindelemente des gegenwärtigen Knotens
child::absatz	Alle absatz-Kindelemente des gegenwärtigen Knotens
child::text()	Alle Textknoten des gegenwärtigen Knotens
child::node()	Alle Kindknoten des gegenwärtigen Knotens
attribute::name	Das Attribut name
attribute::*	Alle Attribute des gegenwärtigen Knotens
descendant::absatz	Alle absatz-Nachfahren des gegenwärtigen Knotens
/descendant::absatz	Alle absatz-Elemente in diesem Dokument
/descendant::kapitel/child::absatz	Alle absatz-Kindelemente, deren direkter Vorfahr kapitel ist
descendant-or-self::absatz	Alle absatz-Nachfahren des gegenwärtigen Knotens oder er selbst (wenn er ein absatz ist)
ancestor::absatz	Alle absatz-Ahnen des gegenwärtigen Knotens
ancestor-or-self	Alle absatz-Ahnen des gegenwärtigen Knotens oder er selbst
self::absatz	Der Knoten selbst, wenn es sich um einen absatz handelt

child::kapitel/descendant::absatz	Alle absatz-Elemente der kapitel-Kindelemente
child::*/child::absatz	Alle absatz-Kindelemente der direkten Kindelemente des gegenwärtigen Knotens (Enkel-Absätze)
child::absatz[position = 1]	Erstes absatz-Kindelement des gegenwärtigen Elements
child::absatz[position = last()]	Letztes absatz-Kindelement des gegenwärtigen Elements
child::absatz[position = last() - 1]	Vorletztes absatz-Kindelement des gegenwärtigen Elements
following-sibling::kapitel[position() = 1]	Das dem gegenwärtigen Knoten nächstliegende Element kapitel (nachfolgend)

Core Function Library
Kernfunktionen für den Datenzugriff

- Spezifikation von Xpath sieht vor, daß Implementierungen eine Reihe von Funktionen beinhalten müssen. (Core Funktion Library)
- Funktionen sind die gleichen die in XSLT ständig benötigt werden.
- **Wichtig**: Befehle sind noch nicht „final“ bzw. dies sind nicht alle Befehle die XSLT anbieten wird, da XSLT 1.0 noch nicht als „fertig“ angesehen werden sollte.

Prädikaten- Funktionen

Funktionsname	Rückgabewert	Rückgabetyp
Knotensatz-orientierte Funktionen		
`last()`	Anzahl (üblicherweise von Elementen in einem Kontext)	`number`
`position()`	Kontext-Position	`number`
`count(node-set)`	Anzahl der Knoten innerhalb des `node-set`	`number`
`id(object)`	Element mit dem Attribut `id` vom Wert `object`	`node-set`
Mehrere dieser Funktionen haben das optionale Argument `node-set`. Fehlt es, heißt das in allen Fällen, dass der gegenwärtige Knoten als Argument zählt.		
Zeichenorientierte Funktionen		
`string(object?)`	Konvertiertes Objekt als Zeichenkette	`string`
`concat(string, string, string*)`	Die Aneinanderreihung der Argumente	`string`
`starts-with(string, string)`	Wahr, wenn der die erste Zeichenkette mit der zweiten beginnt (sonst unwahr)	`boolean`
`contains(string, string)`	Wahr, wenn die erste Zeichenkette die zweite beinhaltet (sonst unwahr)	`boolean`

Boolesche Funktionen		
`boolean(object)`	Das in einen boolschen Wert konvertierte Argument	`boolean`
`not()`	Wahr, wenn das Argument unwahr ist (sonst unwahr)	`boolean`
`true()`	Wahr	`boolean`
`false()`	Unwahr	`boolean`
Numerische Funktionen		
`number(object?)`	Der in `number` konvertierte Wert des Objekts (Zeichenkette, boolscher Wert, Knotensatz oder Objekt); fehlt das Argument, gilt der gegenwärtige Knoten	`number`
weitere wie sum,floor...		

Änderungen in XPath 2.0

- Knotenmengen werden durch Sequenzen ersetzt. Eine Sequenz ist (geordnete) Folge von Knoten oder Werten anderer Datentypen.
- Zum gewöhnlichen Vereinigung- gibt es jetzt auch Durchschnitt- und Differenzoperatoren.
  - Beispiel.:
  - Vereinigung: (K, L, M) union (L, M, N) ist (K, L, M, N)
  - Durchschnitt: (K, L, M) intersect (L, M, N) ist (L, M)
  - Differenz: (K, L, M) except (L, M, N) ist (K)
- Diese Operatoren können auch in Pfadausdrücken eingesetzt werden.

Änderungen in XPath 2.0

- To-Operator: liefert längere Sequenzen ganzer Zahlen.
  - Beispiel: 1 to 10 // liefert die zahlen von 1 bis 10 in einer Sequenz.
- For-Operator: ermöglicht den Zugriff auf einzelne Sequenzelemente, aus denen man neue Sequenzen bildet-
  - Beispiel: For $x in (1 to 10) return $x * $x // liefert die Quadratzahlen von 1 bis 10.

Änderungen in XPath 2.0

- Quantifizierung können direkt angegeben werden.

- Beispiele:

some $a in (1,2,3), $b in (3,4,5) satisfies $a + $b = 6

ist true, weil z.B. 1+5=6

every $a in (1,2,3), $b in (3,4,5) satisfies $a + $b = 6

ist false, weil z.B. 1+3 != 6

BEISPIELE

- Und jetzt kommen die Beispiele...

Fazit

+ Umfangreiche Funktionsbibliothek
+ Unterstützt XML-Schema
+ Leichter Umgang da viele Begriffe aus anderen Programmiersprachen bekannt.

- Lange Ausdrücke bei unverkürzten Pfaden.

Literaturhinweise und Quellen

- Behme, Henning und Mintert, Stefan: XML in der Praxis.
- Kränzler, Christine: XML/XSL für Buch und Web
- Kay, Michael: XPath 2.0 : programmer‘s reference.
- Bach, Mike: XSL und XPath - verständlich und praxisnah.

Literaturhinweise und Quellen

- Simpson, John E.: XPath and XPointer.
- Skonnard, Aaron und Gudgin, Martin: Essential XML Quick Reference.
- Becker, Oliver: Navigation in XPath. Java Spectrum 6/2003
- www.W3C.org
- de.Wikipedia.org