

Въведение в XSLT (eXtensible Stylesheet Language for Transformations), XPath и XQuery



Преглед на XSL
Употреба
Основи на XPath
Синтаксис
Локации и оси
XQuery
Примери

Литература

- Jonathan Pinnock, et al. “**Professional XML, 2nd edition**”, ISBN: 1861005059, WROX Publishers, 2001
- Serge Abiteboul, Peter Buneman and Dan Suciu, “**Data on the Web: from Relations to Semistructured Data and XML**”, ISBN 1-55860-622-X, Morgan Kaufmann Publishers, 2000
- World Wide Web Consortium, “**XQuery 1.0. An XML Query Language**”, W3C Working Draft, Apr. 30, 2002
- World Wide Web Consortium, “**XML Path Language (XPath) Version 1.0**”, W3C Recommendation, Nov. 16, 1999
- **Qexo: The GNU Kawa implementation of XQuery**, <http://www.gnu.org/software/qexo/>
- Don Chamberlin, Jonathan Robie, and Daniela Florescu, “**Quilt: An XML Query Language for Heterogeneous Data Sources**”, WebDB 2000, Dallas, May 2000

Стилови множества (Style Sheets)

Какво е style sheet?

- 1. a manual detailing the house style of a particular publisher, publication, etc.
- 2. [COMPUTING] a type of template file consisting of font and layout settings to give a standardized look to documents.

Източник: <https://languages.oup.com/google-dictionary-en>

По-точно:

- Начин на задаване на визуалното представяне на маркирано съдържание в дадена медия (напр. браузър или интерактивна телевизия)

Езици за стилови множества (Style Sheets)

- ***CSS – Cascading Style Sheet Specification***

- Предоставя прост не-XML синтаксис за добавяне на стилове към елементи (в HTML браузъри)

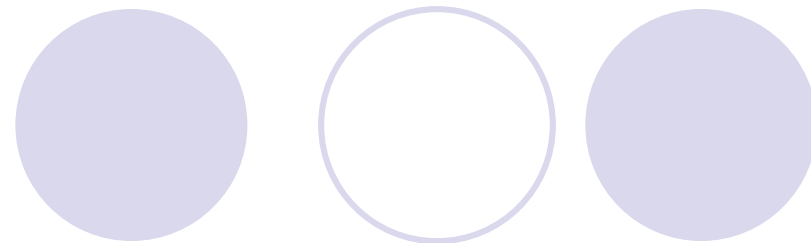
- ***DSSSL – Document Style and Semantics Specification Language***

- Международен SGML стандарт за стилове и конвертиране на документи

- ***XSL – Extensible Stylesheet Language***

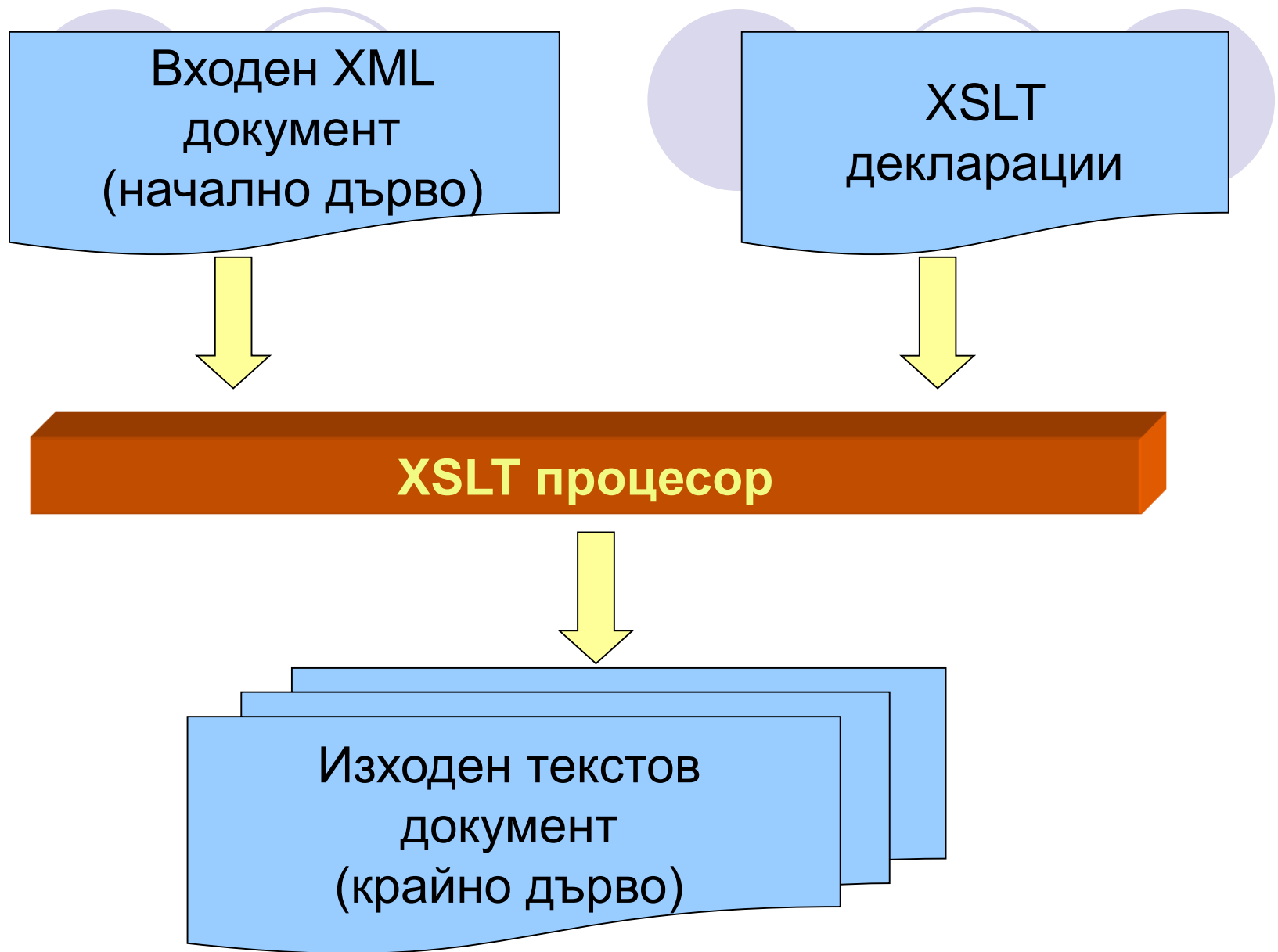
- Комбиниращи черти на DSSSL и CSS, използвайки XML синтаксис

XSL-FO и XSLT



XSL се състои от две части:

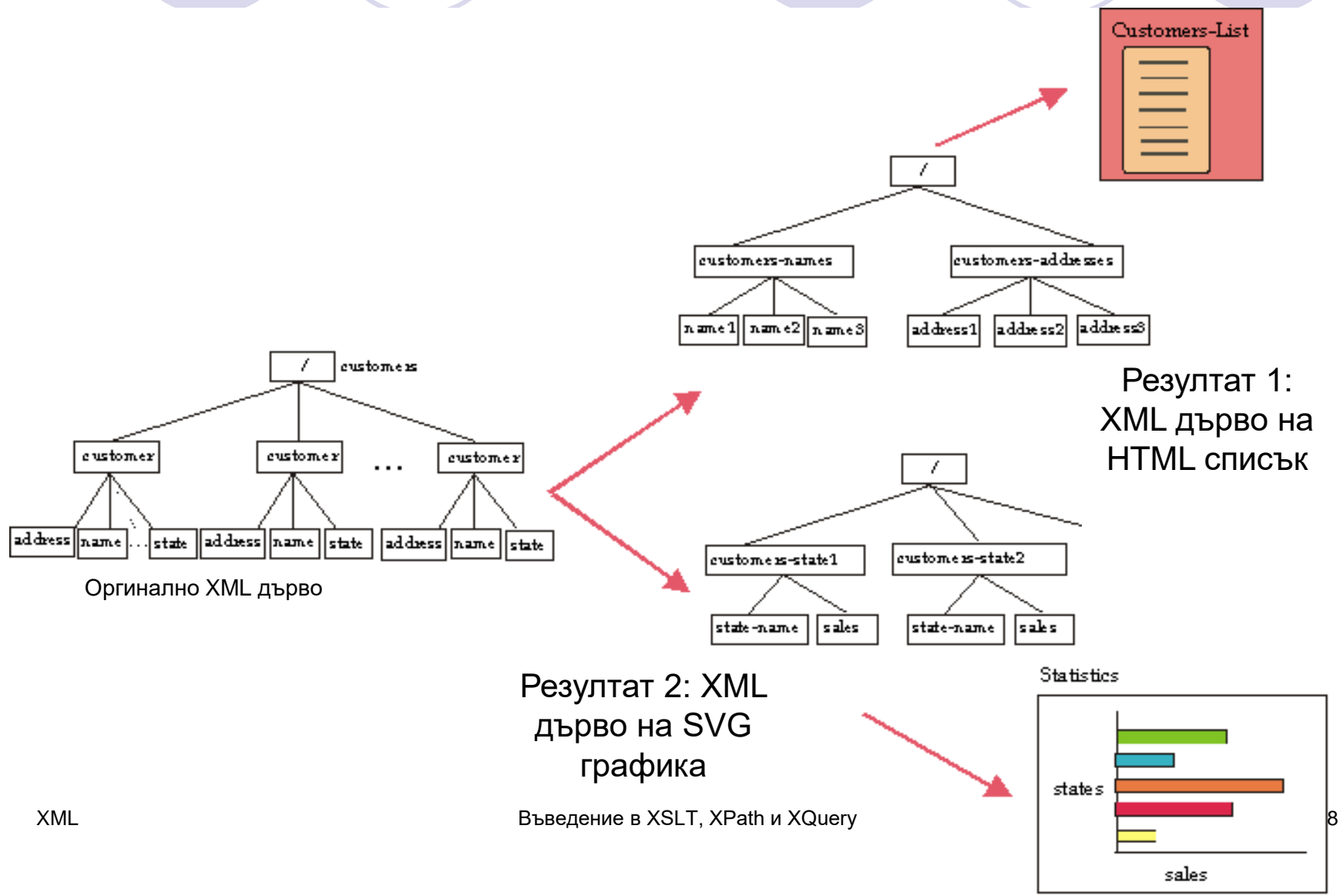
- **Extensible Stylesheet Language – Formatting Objects (XSL-FO)** – език за описание на форматирането на данните в XML документ с цел представянето им на различни медии (напр. екран, принтер или мултимедия);
- **Extensible Stylesheet Language for Transformation (XSLT)** – за трансформиране на XML документи с помощта на различни стилове и функции. Най-често се използва за конвертиране на документ от XML формат към документ в HTML формат, обикновен текст или пък например друг XML документ. Полезен е, когато искаме да разделим презентационния слой на едно приложение от модела на данните му



Възможности на XSL/XSLT

- Добавяне на префиксен или суфиксен текст към съществуващо съдържание
- Структурни промени на входното съдържание, като създаване, отстраняване, редактиране, пренареждане и сортиране на XML елементи
- Многократно използване на елементно или атрибутно съдържание на друго място в документа
- Трансформиране на данни между XML формати
- Определяне на XSL форматиращи обекти и на други средства (напр. CSS) за представяне на съдържанието в дадена медиа (напр. браузър или на хартия), с цел да се прилагат към даден елемент

Използване на XSL/XSLT



Прост пример

- **Файл data.xml:**

```
<?xml version="1.0"?>  
  <?xml-stylesheet type="text/xsl" href="render.xsl"?>  
  <message>Hmmm...</message>
```

- **Файл с дефиниция на трансформация render.xsl:**

```
<?xml version="1.0"?>  
  <xsl:stylesheet version="1.0"  
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">  
    <!-- one rule, to transform the input root (/) -->  
    <xsl:template match="/">  
      <html><body>  
        <h1><xsl:value-of select="message"/></h1>  
      </body></html>  
    </xsl:template>  
  </xsl:stylesheet>
```

Файл с разширение .xsl

- Всеки XSLT документ има .xsl разширение
- XSLT документ започва с:

```
<?xml version="1.0"?>  
<xsl:stylesheet version="1.0"  
    xmlns:xsl="http://www.w3.org/1999/  
        XSL/Transform">
```
- Може да съдържа шаблони, като напр.:

```
<xsl:template match="/"> ... </xsl:template>
```
- Завършва с:

```
</xsl:stylesheet>
```

Намиране на текста message

- Шаблонът `<xsl:template match="/">` задава селектиране на целия входен документ, т.е. на *root възела на XML дървото*
- Вместо това,
 - `<xsl:value-of select="message"/>` селектира преките наследници на message
 - Това са XPath изрази, както и аналогичните им:
 - `./message`
 - `/message/text()` (`text()` е **XPath функция**)
 - `./message/text()`

Как работи ?

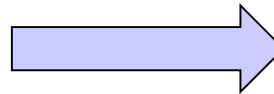
- XSL дефиницията **render.xsl** е:

```
<xsl:template match="/">  
  <html><body>  
    <h1><xsl:value-of select="message"/></h1>  
  </body></html>  
</xsl:template>
```
- Шаблонът **<xsl:template match="/">** избира корена
- **<html><body>** **<h1>** се записва в изходния файл
- Съдържанието на **message** се записва в изходния файл
- **</h1>** **</body></html>** се записва в изходния файл

Вход:

```
<?xml version="1.0"?>  
<?xml-stylesheet type="text/xsl"  
href="render.xsl"?>  
<message>Hmmm...</message>
```

XML



Резултат:

```
<html><body>  
  <h1>Hmmm...</h1>  
</body></html>
```

XSLT спецификация

- Налична на <http://www.w3.org/XSL/>
 - Дефинира 34 елемента и техните атрибути
 - Изисква:
 - **Namespaces**
 - **XPath** (≠**expat**, i.e. expatriate)

XPath и XQuery

Съществуват два популярни декларативни езици с не-XML синтаксис + технологии, предназначени за адресиране (локализиране) на определени части и структури от XML документ:

- XPath се използва за адресиране и манипулиране на секции от XML документ:
 - много популярен стандарт от 1999 година насам
 - използва се от останалите XML спецификации XPointer, XQL, XSLT. Локацията се задава чрез URL
 - работи с интернет, интранет и файловата система
- XQuery (от 2007 год. насам) е друг език за описание на заявки към XML документ, но подобно на SQL заявките към релационна база от данни.

XPath модел



- В XPath 1.0 модела, повечето съставни части от XML документа са представени като възли, свързани с определени отношения.
- Коренът на XPath 1.0 дървото представлява самият документ, а не коренът на документа.
- Всеки елемент в XML документа се представя от елементен възел в дървото.
- Всеки атрибут се представлява от атрибутен възел и по аналогичен начин се представят коментари и инструкции за обработка.
- Текстовият възел представя текстово съдържание на елемент.
- Използваните в документа пространства от имена са представени от възли от тип пространство

Исходен документ

```
<?xml version="1.0" encoding="UTF-8"?>
<network>
  <description name="Boston">
    This is the configuration of our network in the
    Boston office.
  </description>

  <host name="agatha" type="server" os="linux">
    <interface name="eth0" type="Ethernet">
      <arec>agatha.example.edu</arec>
      <cname>mail.example.edu</cname>
      <addr>192.168.0.4</addr>
    </interface>
    <service>SMTP</service>
    <service>POP3</service>
    <service>IMAP4</service>
  </host>

  <host name="gil" type="server" os="linux">
    <interface name="eth0" type="Ethernet">
      <arec>gil.example.edu</arec>
      <cname>www.example.edu</cname>
      <addr>192.168.0.5</addr>
    </interface>
    <service>HTTP</service>
    <service>HTTPS</service>
  </host>

  <host name="baron" type="server" os="linux">
    <interface name="eth0" type="Ethernet">
      <arec>baron.example.edu</arec>
      <cname>dns.example.edu</cname>
      <cname>ntp.example.edu</cname>
      <cname>ldap.example.edu</cname>
      <addr>192.168.0.6</addr>
    </interface>
    <service>DNS</service>
    <service>NTP</service>
    <service>LDAP</service>
    <service>LDAPS</service>
  </host>
```

XML

```
<host name="mr-tock" type="server"
  os="openbsd">
  <interface name="fxp0" type="Ethernet">
    <arec>mr-tock.example.edu</arec>
    <cname>fw.example.edu</cname>
    <addr>192.168.0.1</addr>
  </interface>
  <service>firewall</service>
</host>

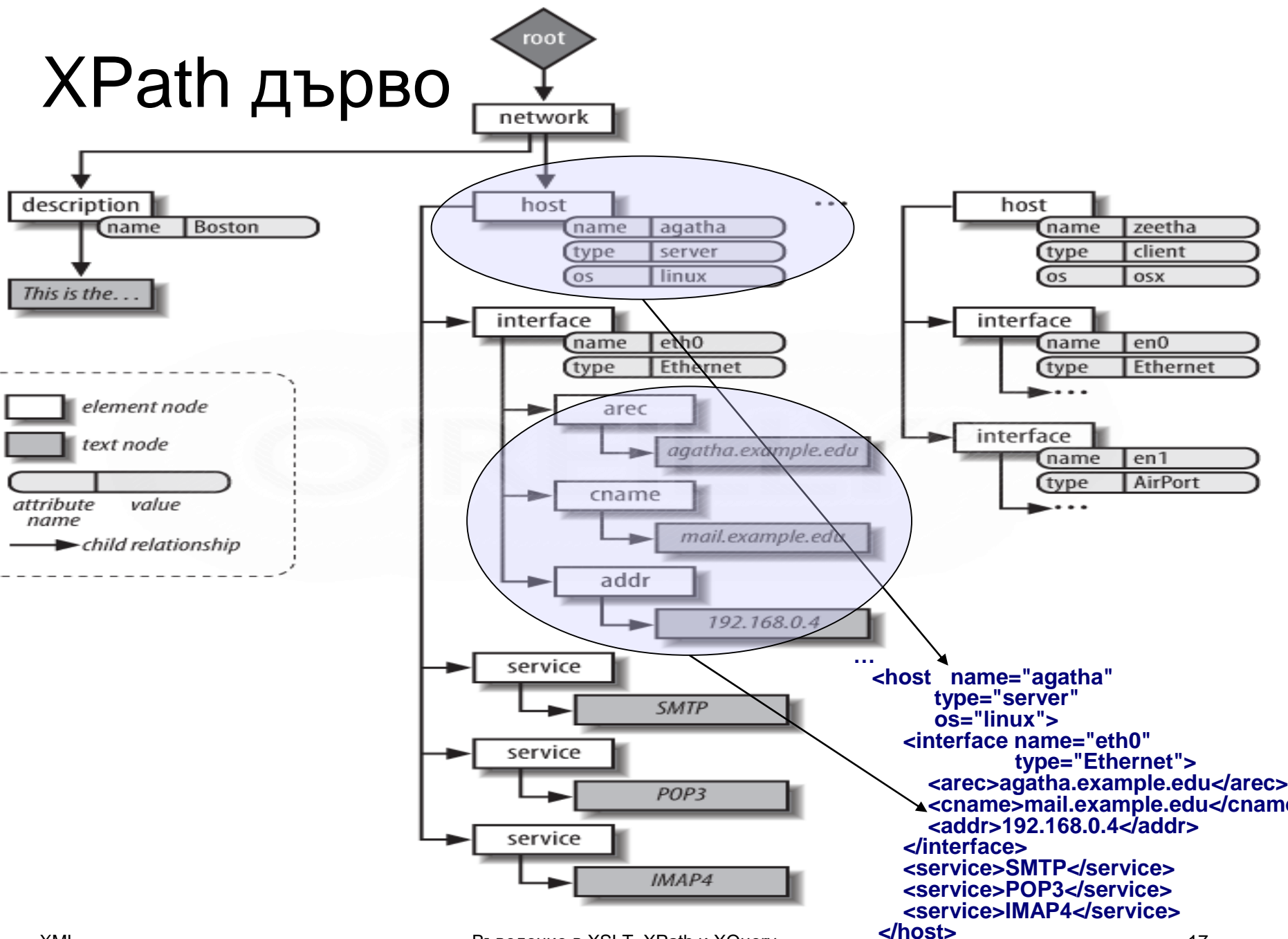
<host name="krosp" type="client" os="osx">
  <interface name="en0" type="Ethernet">
    <arec>krosp.example.edu</arec>
    <addr>192.168.0.100</addr>
  </interface>
  <interface name="en1" type="AirPort">
    <arec>krosp.wireless.example.edu</arec>
    <addr>192.168.100.100</addr>
  </interface>
</host>

<host name="zeetha" type="client" os="osx">
  <interface name="en0" type="Ethernet">
    <arec>zeetha.example.edu</arec>
    <addr>192.168.0.101</addr>
  </interface>
  <interface name="en1" type="AirPort">
    <arec>zeetha.wireless.example.edu</arec>
    <addr>192.168.100.101</addr>
  </interface>
</host>
</network>
```

Въведение в XSLT, XPath и XQuery

16

XPath дърво

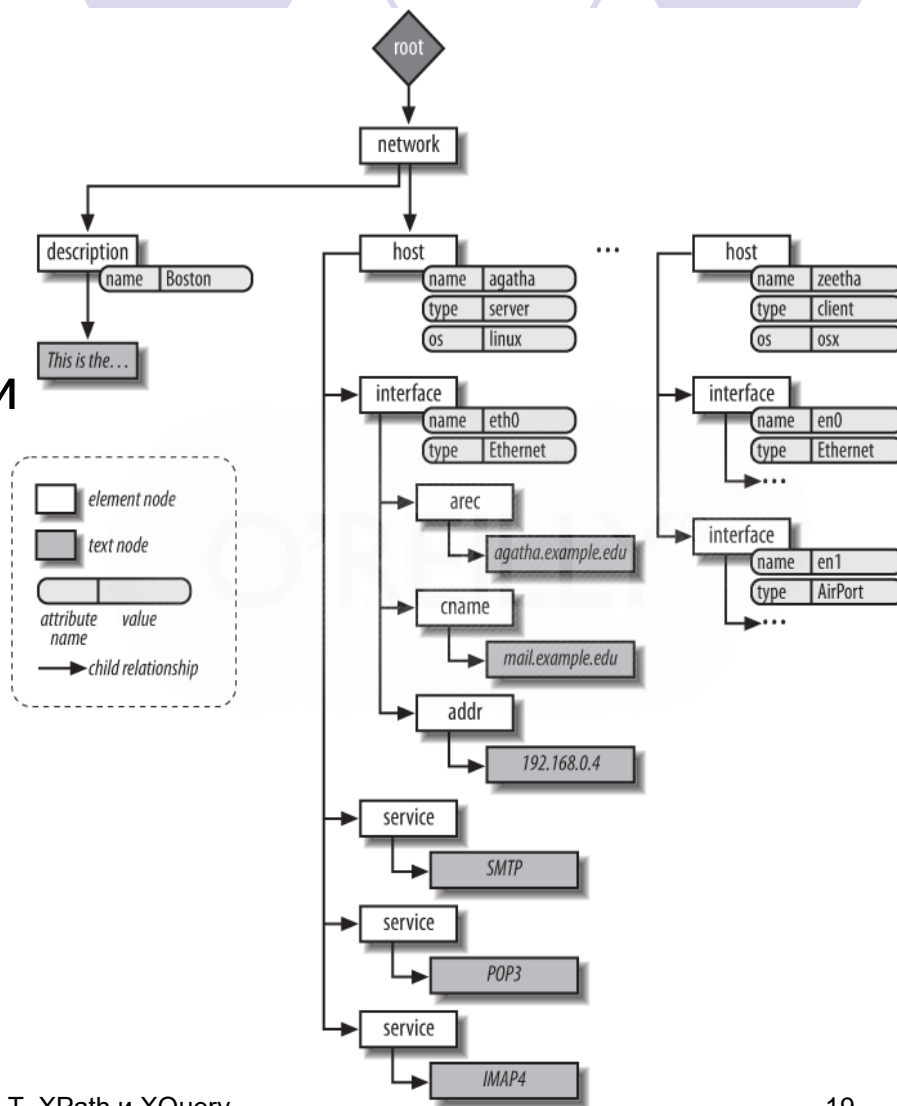


Контекстен възел

- XPath процесорът изчислява XPath израз, който задава път от дадена начална точка в дървото, например от корена.
- Чрез изчислението на този път като последователност от свързани възли, процесорът се „придвижва“ по дървото, моделиращо документа, до указания от израза възел.
- Този възел, където за момента се намира XPath процесорът, се нарича **контекстен възел**.
- От него започват всички относителни пътища от този момент до следващото позициониране на процесора в друг контекстен възел.
- Контекстният възел се задава с ‘.’
- XPath връща множество от възли от дървото, а не XML документ (!)

Видове XPath възли

- възел-корен на дървото, сочещ към корена на документа
- възел-елемент - възелът, представляващ корена на документа, е задължителен (възелът с име network); други възли от този тип са host, interface и service;
- възел-текст - напр. възелът със съдържание "This is the...";
- възел-атрибут - като напр. name, type и os;
- възел-инструкция за обработка;
- възел-декларация на CDATA секция (област от данни);
- възел-коментар.

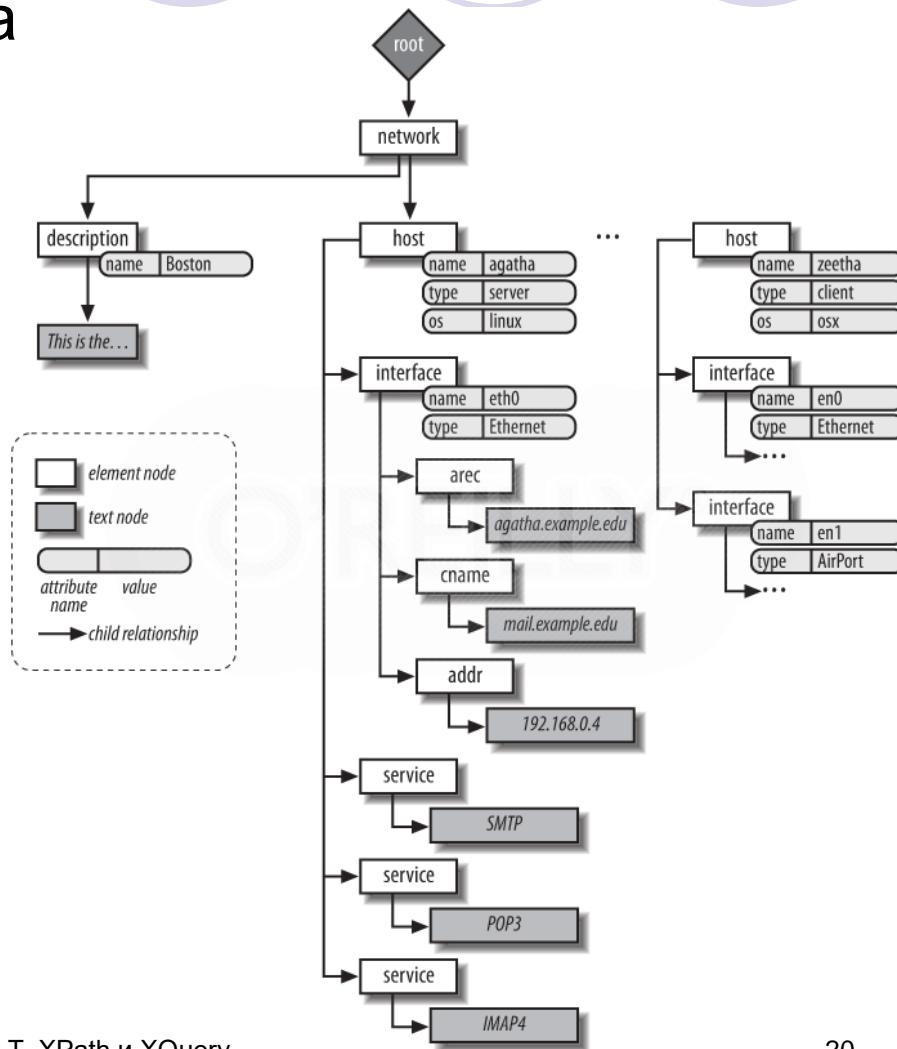


Отношения между възли 1/2

- Възлите в XPath дървото са свързани с отношения от два типа:

1. отношение на свързване на възел с подреден списък от възли-наследници

2. отношение на свързване на възел с неопределено множество от други върхове



Отношения между възли 2/2

В подреден списък се представят:

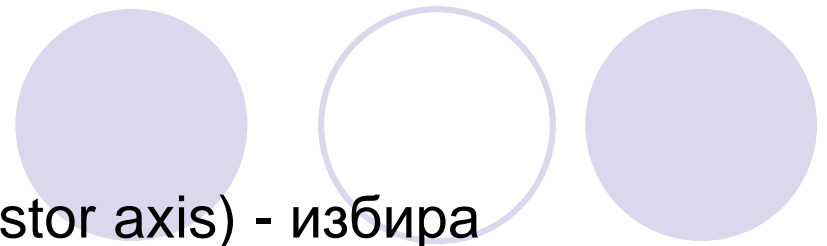
- наследниците на даден елемент (понеже те образуват подредена йерархия на елементното съдържание),
- текстовите възли,
- инструкциите за обработка и
- коментарите

Не са подредени:

- възлите за атрибути и
- CDATA секциите (областите от данни).

Наистина, търсенето на *n*-ти елемент от йерархията от наследници за даден елемент има смисъл, докато търсенето на *n*-ти атрибут на елемент не може да дефинира върху неподредени атрибути. Също така, не се използва търсене на част от CDATA секция.

XPath 1.0 оси 1/3

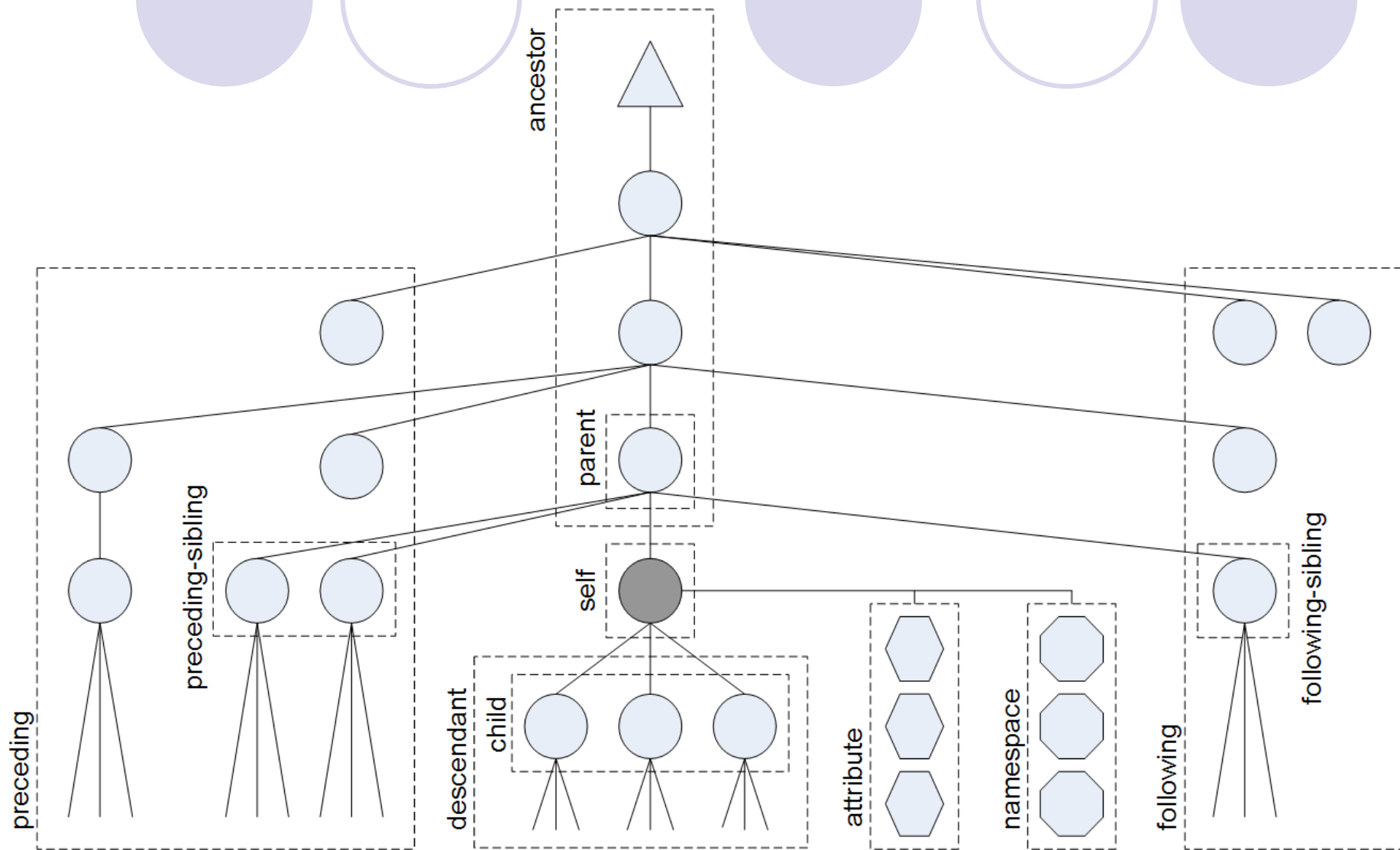


1. Ос на предшествениците (ancestor axis) - избира предшествениците на контекстния възел в обратен ред на появата им в документа
2. Ос на предшествениците и на самия възел (ancestor-or-self axis) - избира контекстния възел и неговите предшественици в обратен ред на появата им
3. Ос на директните наследници, или ос на децата (child axis) – селектира преките наследници на контекстния възел
4. Ос на атрибутите (attribute axis) – задава всички атрибути на контекстния възел
5. Ос на наследниците (потомците) (descendant axis) – връща всички наследници (преки и непреки) на контекстния възел по ред на появата им в документа. Атрибутите и секциите за данни не се разглеждат като наследници на елементен възел
6. Ос на наследниците и на самия възел (descendant-or-self axis) – връща резултата на оста на наследниците плюс контекстния възел

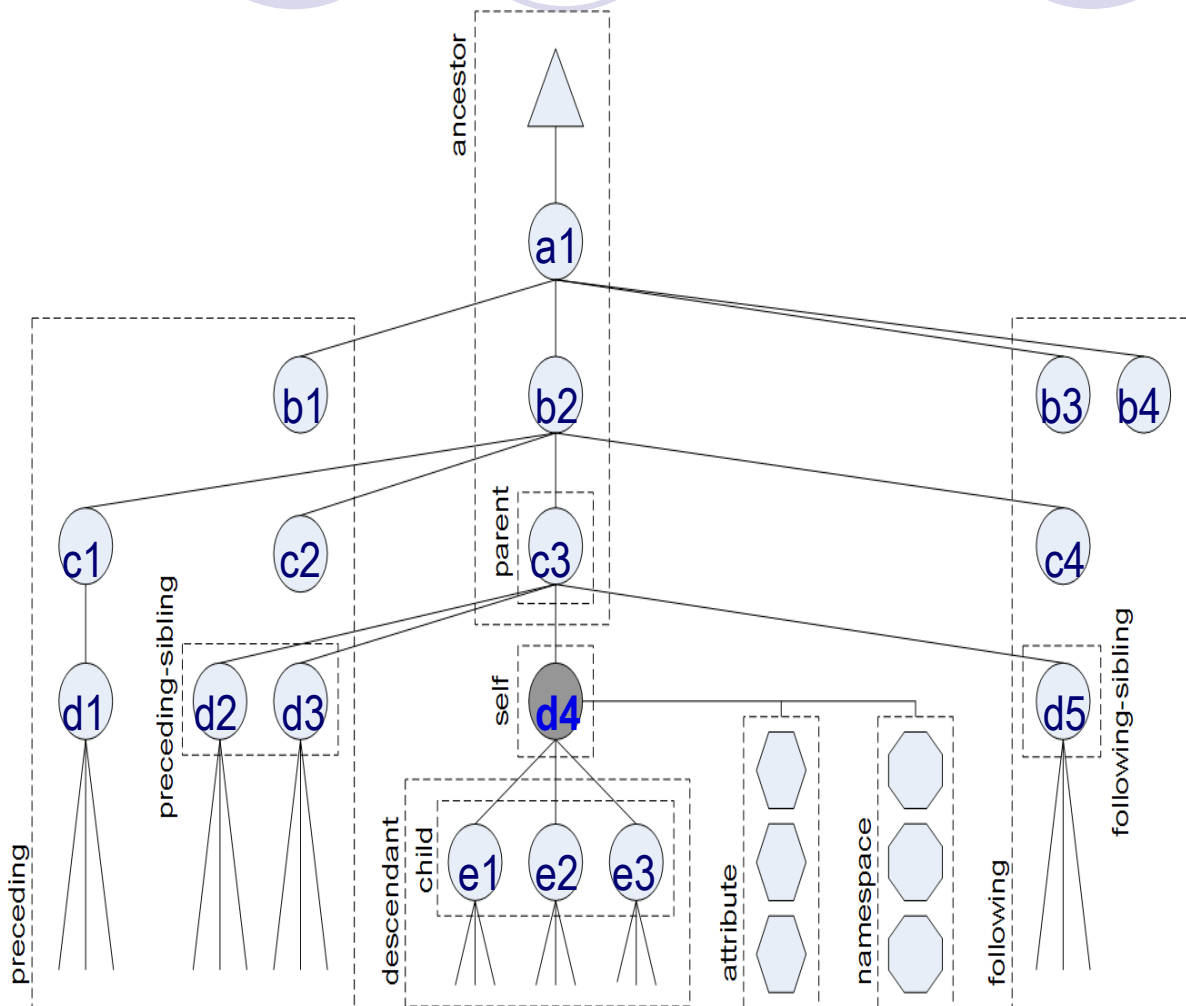
XPath 1.0 оси 2/3

7. Ос на следващите възли (following axis) – селектира всички следващи възли след контекстния възел, без наследниците и атрибутите му и без пространствата от имена
8. Ос на следващите възли и на самия възел (following-sibling axis) – избира всички братя и сестри на контекстния възел, следващи вдясно от него
9. Ос на пространствата от имена (namespace axis) – не се препоръчва в XPath 2.0
10. Ос на родителя (parent axis) – връща родителя на контекстния възел
11. Ос на предходните възли (preceding axis) – селектира всички предходни възли спрямо контекстния възел, без наследниците и атрибутите му, и без пространствата от имена
12. Ос на предходните възли и на самия възел (preceding-sibling axis) – избира всички предшестващи братя и сестри на контекстния възел вляво от него
13. Ос на самия възел или собствена ос (self axis)

XPath 1.0 оси 3/3



XPath 1.0 оси и XML



XML

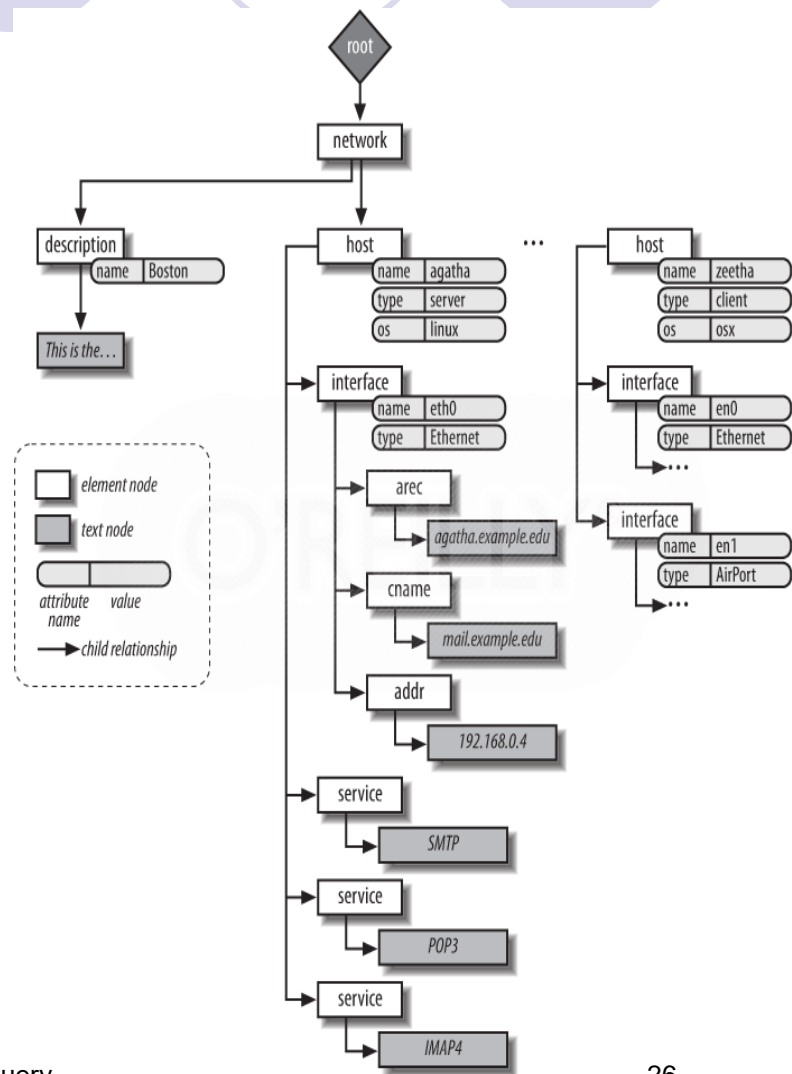
Введение в XSLT, XPath и XQuery

```

<a1>
  <b1/>
  <b2>
    <c1>
      <d1> .... </d1>
    <c1/>
    <c2/>
    <c3>
      <d2> .... </d2>
      <d3> .... </d3>
      <d4 attr1='...', ..., attr3='...'>
        <e1> .... </e1>
        <e2> .... </e2>
        <e3> .... </e3>
      </d4>
      <d5> .... </d5>
    </c3>
  <c4/>
</b2>
<b3/>
<b4/>
</a1>
  
```

Относителни и абсолютни пътища

- **Относителният път** на местоположение се състои от една или повече стъпки на местоположение (*location step*), разделени със знака '/
- **Абсолютният път** на местоположение се състои от знака '/', последван от относителен път.
- **./network/host** е абсолютен път на местоположение и задава възела на първия елемент с име **host** за корена на документа.
- Относителните пътища могат да съдържат в началото си низа './' - напр. **./interface/cname**



Разширен синтаксис 1/2

axis :: node-test [predicate] ... [predicate]

- Ос - индикатор за връзка между текущия възел, избран на предходната стъпка, и избраните от стъпката възли.
- Филтър - тест за възела (*node test*) - задава характеристики на възлите, които ще бъдат избрани при удовлетворяване на условията:
 1. Задаването на името на възел като филтър тества за възли с това име, тоест избира всички възли с това име. Така например **child::elementName** връща всички наследници (деца) на контекстния възел, които са с име **elementName**
 2. Във филтрите могат да се ползват специфични функции за тестване на типа на възела на избраната ос. Ако функцията върне стойност истина, то възелът ще бъде избран:
 1. Тестът **text()** връща истина за всеки текстов възел, затова **child::text()** ще избере само текстовите възли - деца на контекстния възел.
 2. **comment()** връща истина за всеки възел-коментар, така че **child::comment()** ще избере само тези деца на контекстния възел, които са от тип коментар.
 3. **processing-instruction()** връща истина за всеки текстов възел - инструкция за обработка.
 4. Тестът **node()** връща истина за всеки възел от всякакъв тип, затова **self::node()** избира контекстния възел, тоест '.'

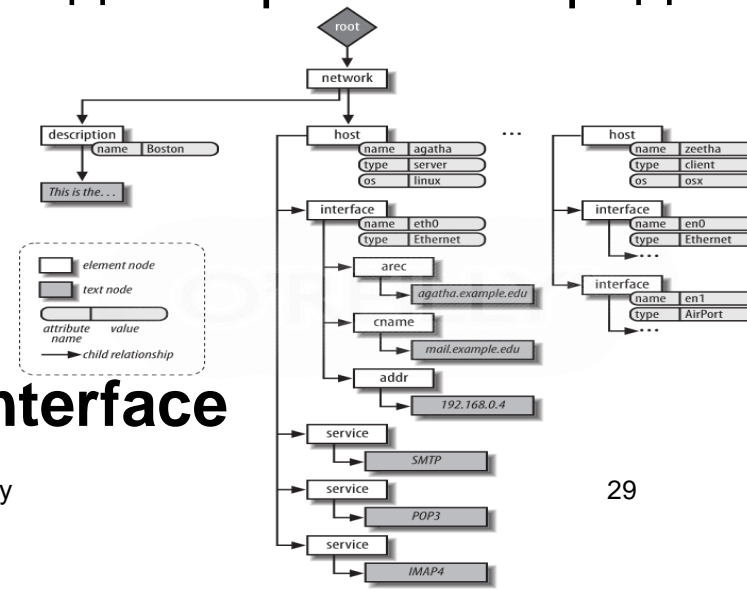
Разширен синтаксис 2/2

```
axis :: node-test [ predicate ] ... [predicate]
```

- Предикатите, ако такива съществуват, филтрират допълнително набора от възли (върнат от филтъра) по отношение на дадена ос спрямо контекстния възел.
- Всеки предикат се изчислява за всеки възел от набора от възли и ако върне стойност истина, този възел се добавя към резултатния набор от възли. Така например:
 - **descendant::document[attribute::level = "confidential"]**
 - задава елементите-наследниците (те могат да бъдат само елементи, защото само елементи изграждат йерархията на дървото) на контекстния възел, които са с име document и с атрибут с име level, имащ стойност "confidential".
 - **preceding::*** - адресира всички предшественици на контекстния възел чрез използване на заместващия символ "*", който задава елемент с какво и да е име.

Примери с разширен синтаксис 1/2

- Допълнително отношение в предиката може да има позицията на близостта (*proximity position*) спрямо дадена ос
- Позицията на близостта на член на набора възли по отношение на дадена ос се определя от номера на позицията на възела в подредените в документа възли
- Например **pElement[position()=3]** задава третият по ред възел с име **pElement**.
- Преминаването през елементите йерархията отдясно:



child::network/child::host/child::interface

Примери с разширен синтаксис 2/2

- Друг пример:
child::chapter/descendant-or-self::node()/child::section
- - при изчислението му процесорът ще премине през цялата йерархията на елемента **chapter** и избере всички елементи с име **section**
- Изразът **descendant-or-self::node()** работи рекурсивно по йерархията на наследниците и е мощно средство за претърсване на дървета
- **child::para[attribute::type='warning'][position()=2]**
- - селектира втория пряк наследник **para** на контекстния възел, който има атрибут с име **type** и стойност **warning**

Кратък синтаксис



- ❖ Очевидно разширеният синтаксис не е удобен за работа и затова XPath използва и кратък синтаксис
- ❖ Краткият запис на пътищата наподобява добре познатото адресиране по директориите на файловите системи и използва същия синтаксис.
- ❖ Краткият запис използва съкращения като '@' (означение за атрибут) и '*' (избор на всички възли).

Предшественици

- Избор на родителя на контекстния възел:
 - `'..'` или разширената форма
 - `parent()`
- Избор на всички `'title'` елементи – деца на родителя на контекстния възел:
 - `../title` или разширената форма
 - `parent::node()/child::title`

Абсолютни пътища

- Чрез '/' означаваме стартирането от корена (root element)
 - За да намерим всички 'para' елементи в даден документ, ползваме:
 - `//para` или разширената форма
 - `/descendant-or-self::node()/para`

Оператор на рекурсивния спуск

- Рекурсивно търсене сред наследените възли
 - **chapter//para** се спуска в йерархията на chapter и избира всички **para** елементи

- ```
<chapter> //Starting node
 <title>...</title>
 <para>...</para> //Selected
 <note>
 <para>...</para> //Selected
 </note>
</chapter>
```

- Разширено: **child::chapter/descendant-or-self::node() / child::para**

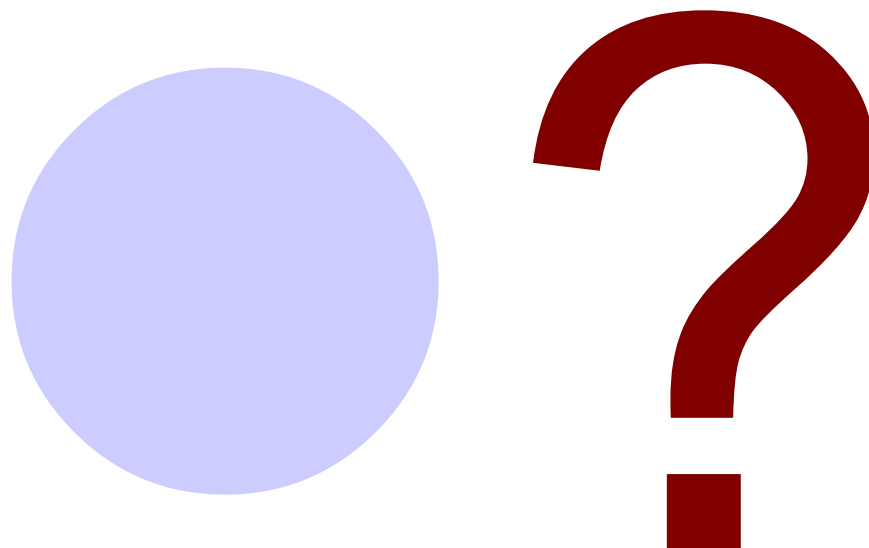
- Избиране на всички **para** елементи от текущия възел: **./para**

- Разширено:  
**self::node() / descendant-or-self::node() / child::para**

# Примери

№	Разширен синтаксис	Кратък синтаксис
1	<b>self::node()/descendant-or-self::node()/child::p</b>	<b>./p</b>
2	<b>parent::node()/child::title</b>	<b>../title</b>
3	<b>child::host[service]</b>	<b>host[service]</b>
4	<b>/child::chapter/descendant-or-self::node()/child::section</b>	<b>/chapter//section</b>
5	<b>attribute::*</b>	<b>@*</b>
6	<b>child::*</b>	<b>*</b>
7	<b>self::node()/child::section[position()=last()]</b>	<b>./section[last()]</b>
8	<b>/child::doc/child::chapter[position()=5]/child::section[position()=2]</b>	<b>/doc/chapter[5]/section[2]</b>
9	<b>parent::node()/attribute::lang</b>	<b>../@lang</b>
10	<b>//descendant-or-self::node()/child::host[attribute::type="server"]</b>	<b>//host[@type="server"]</b>

Какво означава всеки от изразите от  
предния слайд?



# Отговори 1/2

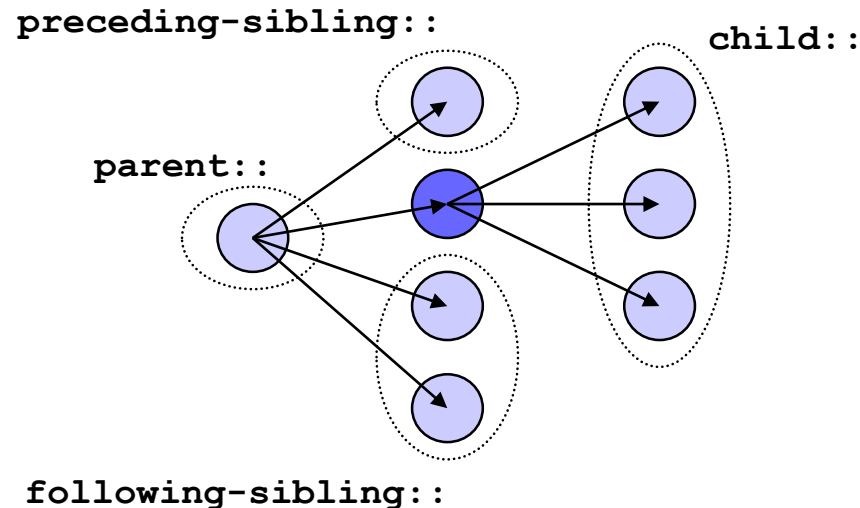
1. **self::node()/descendant-or-self::node()/child::p** или **./p** - задава всички възли с име **p** в йерархията на контекстния възел, т.е. всички негови преки или непреки наследници с това име;
2. **parent::node()/child::title** или **../title** - селектира всички възли с име **title**, които са преки наследници (деца) на родителя на контекстния възел;
3. **child::host[service]** или **host[service]** - избира децата с име **host** на контекстния възел, които имат поне едно дете с име **service**
4. **/child::chapter/descendant-or-self::node()/child::section** или **/chapter//section** - задава всички възли **section** в йерархията на контекстния възел, които имат за предшественик **chapter**
5. **attribute::\*** или **@\*** - избира всички атрибути на КОНТЕКСТНИЯ ВЪЗЕЛ

# Отговори 2/2

6. **child::\*** или **\*** - избира всички деца на контекстния възел
7. **self::node()/child::section[position()=last()]** или **./section[last()]** - селектира последното дете на КОНТЕКСТНИЯ ВЪЗЕЛ с име **section**
8. **/child::doc/child::chapter[position()=5]/child::section[position()=2]** или **/doc/chapter[5]/section[2]** - избира втората секция на петата глава на документа
9. **parent::node()/attribute::lang** или **../@lang** - избира атрибута **lang** на родителя на контекстния възел
10. **//descendant-or-self::node()/child::host[attribute::type="server"]** или **//host[@type="server"]** - избира всички потомци на корена на документа от фиг. от слайд 16/17, които имат име **host** и атрибут **type** със стойност **server**

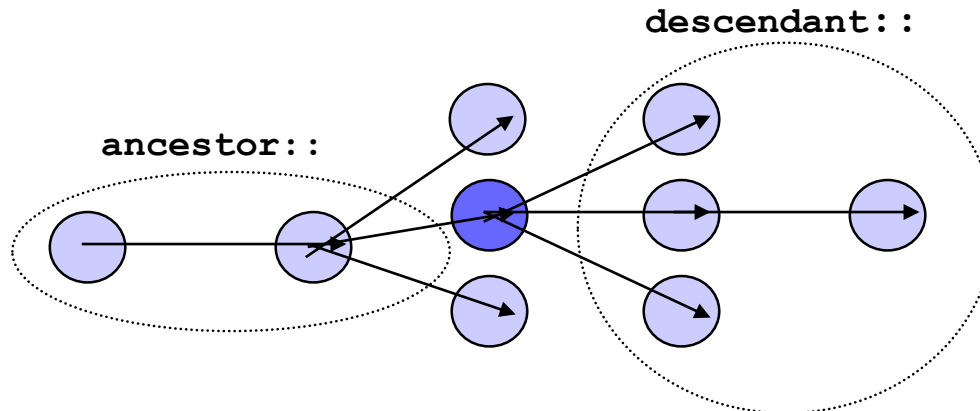
# Други релации 1/4

- Избор на братя и сестри (siblings) на текущия (контекстния) елемент:
  - `preceding-sibling::`
  - `following-sibling::`



## Други релации 2/4

- Избор на всички предшественици (ancestors) и на потомци (descendants) на текущия (контекстния) елемент:
  - `ancestor::`
  - `descendant::`
- (без братя и сестри – siblings)



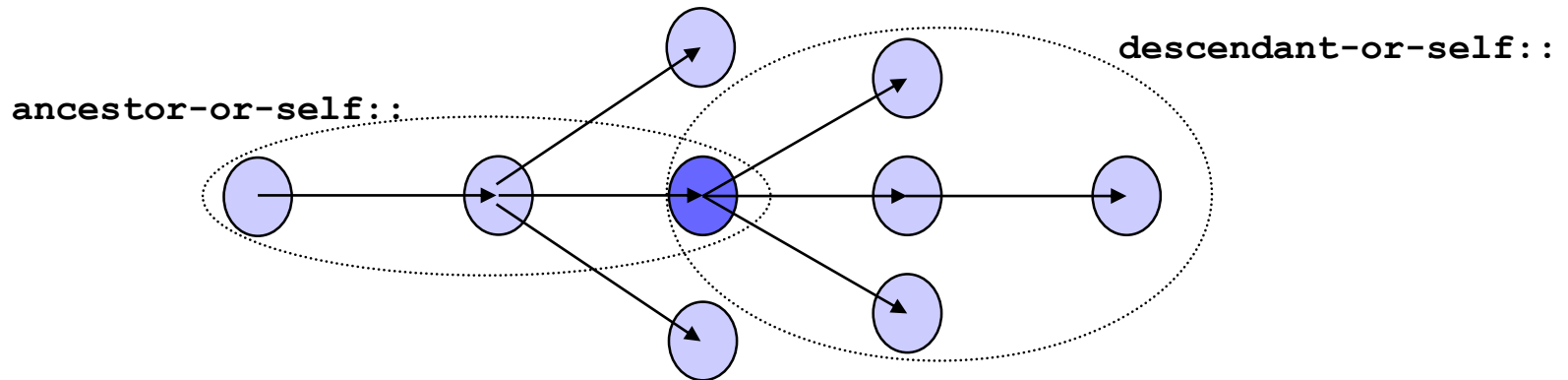


# Други релации 3/4

- Избор на всички предшественици (ancestors) и на потомци (descendants) на текущия (контекстния) елемент, вкл. и самия него:

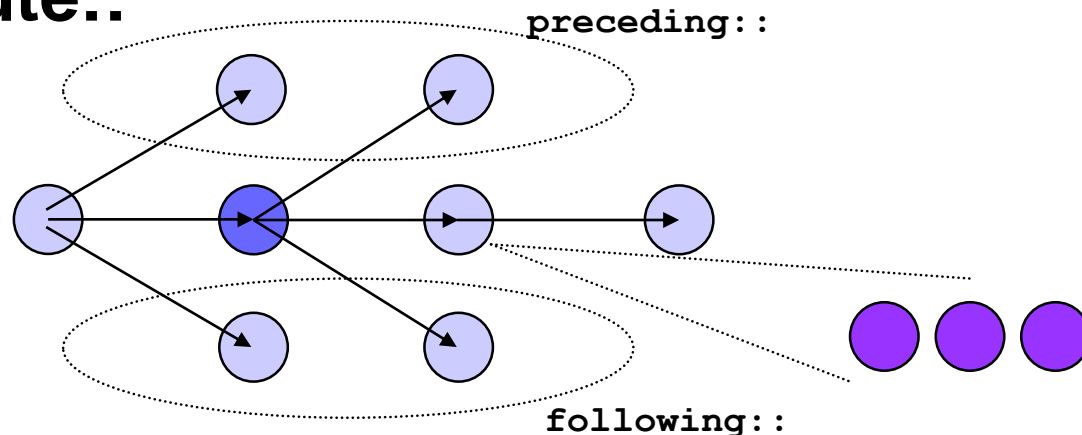
○ **ancestor-or-self::**

○ **descendant-or-self::**



# Други релации 4/4

- Избор на всички предишни и следващи възли на текущия (контекстния) елемент:
  - **preceding::**
  - **following::**
- Избор на атрибутите му:
  - **attribute::**



# Тестове

- Функция **position()**
  - `<xsl:template match="first/second[position() = 2]">` е както
  - `<xsl:template match="first/second[2]">`
- Функции **first()** / **last()**
  - Избор на първи/последен *sibling* в списък
- Функция **count()**
  - Изчислява броя на елементите в списък
  - `child::transcript[count(child::intron) = 1]`
- Функция **id()**
  - Проверява идентификатора на елемента
  - `child::transcript[id("ENS0001")]`

# За повече информация – вижте спецификациите

Part	Date	Status	URL
<b>XSL- Formatting Objects ver. 2.0</b>	<b>04.02.2010</b>	<b>Version 2.0 W3C Working Draft</b>	<b><a href="http://www.w3.org/TR/xslfo20/">http://www.w3.org/TR/xslfo20/</a></b>
<b>XSL Transformations (XSLT) ver. 2.1</b>	<b>11.05.2010</b>	<b>Version 2.1 W3C Working Draft</b>	<b><a href="http://www.w3.org/TR/xslt-21/">http://www.w3.org/TR/xslt-21/</a></b>
<b>XML Path Language (XPath) ver. 2.0</b>	<b>23.01.2007</b>	<b>Version 2.0 W3C Working Draft</b>	<b><a href="http://www.w3.org/TR/xpath20/">http://www.w3.org/TR/xpath20/</a></b>

# XPath 2.0

- XML Path Language (XPath) 2.0 е препоръка (Recommendation) на W3C от 14.12.2010 и е налична на адрес <http://www.w3.org/TR/xpath20/>. Наборът от функции на тази спецификация е много по-богат, мощен и по-чувствителен към типа на данните.
- Също така новост в XPath 2.0 са поредиците или последователностите (*sequences*), които заменят познатите ни от XPath 1.0 множества (набори) от възли. Всички XPath 2.0 изрази се изчисляват върху такива поредици, като в тях може да използват променливи.
- XPath 2.0 също използва пътища за местоположение и използването на дефинираните от версия 1.0 оси, с изключение на оста за пространство от имената. В XPath 2.0 тази ос се счита за остаряла и неактуална, но е включена за обратна съвместимост с **XQuery**.

# XPath и XML Query (XQuery)

За адресиране (локализиране) на определени части и структури от XML документ:

- XPath се използва за адресиране и манипулиране на секции от XML документ:
  - много популярен стандарт от 1999 година насам
  - използва се от останалите XML спецификации XPointer, XQL, XSLT и XQuery
- XQuery (от 2007год.) е друг език за описание на заявки към XML документ, но подобно на SQL заявките към релационна база от данни.

*Следващите слайдове са базирани на презентация на Zaniolo, H. Yang, L.-J. Chen и F. Farfán*

# Цели на XQuery

- Увеличаване на количеството информация, съхранявана, обменена и представена като XML
- Възможност за интелигентно търсене на XML-базирани източници на данни
- Ефективно ползване на силата на XML – гъвкаво представяне на много видове информация от различни източници
- XML езикът за заявки трябва да извлича и интерпретира информация от тези различни източници
- Резултат: XQuery (2007г.)

# Изисквания към XML Query

- Експресивна мощност
- Семантика
- Композируемост
- Използване на XML схема с цел валидация
- Манипулиране на програмата

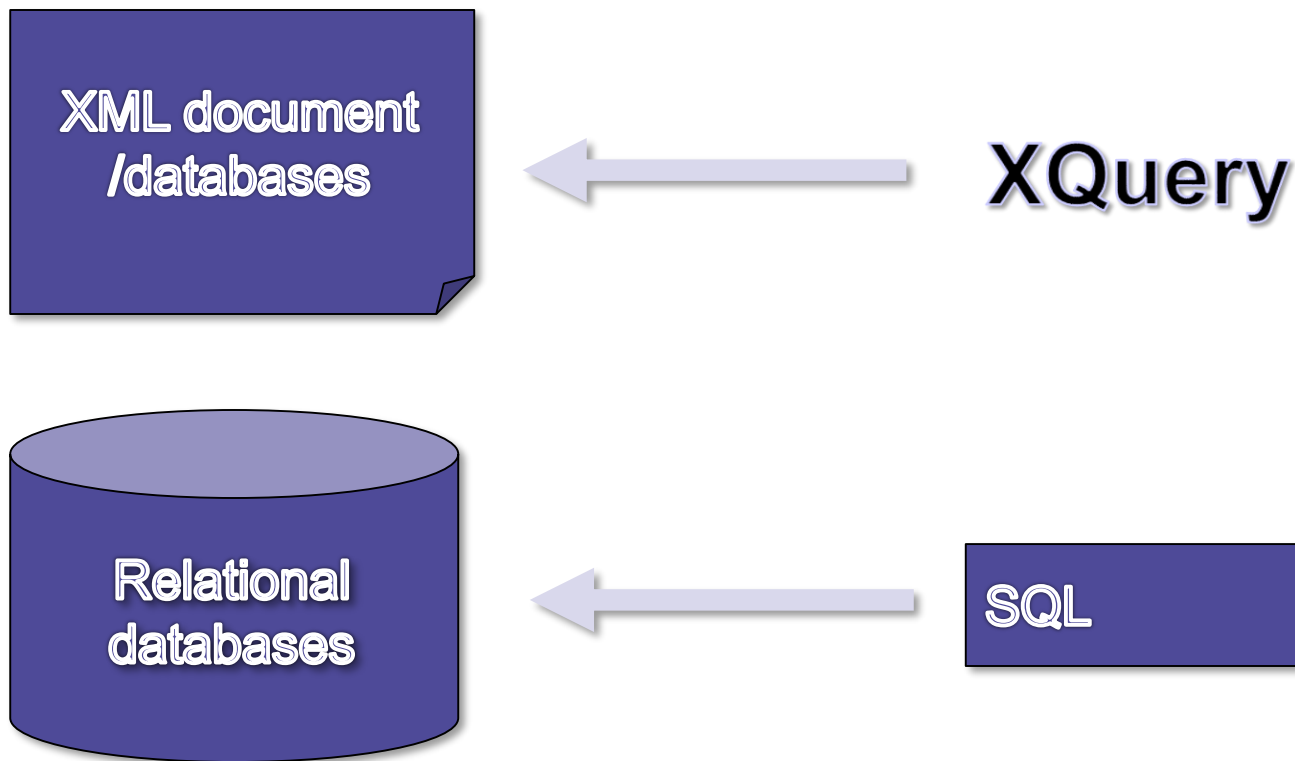


# Езици за заявки



- XPath: синтаксис на израз на път в XML документ, подходящ за йерархични документи
- XML-QL: свързващи променливи (binding variables ) и използване на променливи за създаване на нови структури
- SQL: SELECT-FROM-WHERE модел за реструктуриране на данните
- Quilt: приема много предимства от по-горните езици и е непосредственият предшественик на XQuery

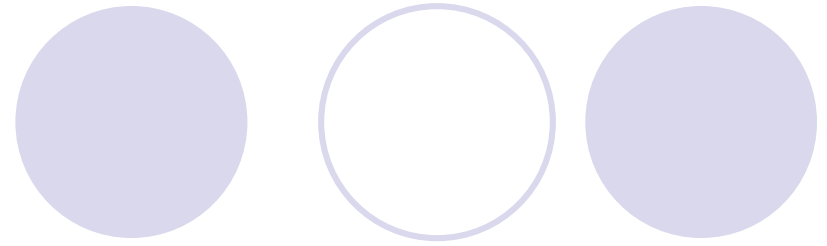
# Използване на XQuery



# XQuery – що е то?

- Създаден според изискванията на W3C XML Query Working Group
  - “XML Query 1.0 Requirements”
  - “XML Query Use Cases”.
- Проектиран да бъде малък и лесно-приложим език
- Достатъчно гъвкав, за да дефинира заявки към широк спектър от XML източници (както бази данни, така и документи).
- Използва синтаксис, който лесно може да се чете от човек

# XQuery – как?



- Ориентиран към изрази - основен градивен елемент
- Функционален език (поне според спецификациите)
- Силно-типизиран (Strongly-typed) език.

# XQuery спрямо XSLT

Преимущества или недостатъци?

- XSLT - document-driven; XQuery - program driven
- XSLT е написан на XML; XQuery – не

Твърдение (да се провери практически 😊):

- с XSLT 2.0 може да се направи всичко това, което може да стане с XQuery

# XQuery - концепции

- Заявката в XQuery е израз, който:
  - Чете различни:
    - XML документи или
    - XML фрагменти
  - Връща последователност от добре оформени (well-formed) XML фрагменти

# Основни форми на XQuery изрази 1/5

- Примитивни (Primary)
  - Литерали, променливи, функционални извиквания и скоби (за задаване на приоритети)
- Път (Path)
  - Открива възли в дърво и връща последователност от отделни възли в реда, зададен в документа
- Последователност (Sequence)
  - Подредена колекция от нула, един или повече елементи, в която елементът може да бъде атомарна стойност или възел
  - Елементът е идентичен на последователност с дължина единица, съдържаща този елемент
  - Последователностите никога не са вложени

# Основни форми на XQuery изрази 2/5

- Аритметични

- Аритметични оператори за добавяне, изваждане, умножение, разделяне и деление по модул

- Условни

- Четири вида сравнения:

- на стойности,
- общи,
- на възли и
- сравнения на заявките

- Логически

- Логическият израз е AND-, NOT- или OR-израз.
- Стойността на логически израз винаги е стойност от тип Boolean



# Основни форми на XQuery изрази 3/5

- Конструктор (Constructor)

- Конструкторите могат да създават XML структури в рамките на заявка
- Има конструктори за елементи, атрибути, раздели CDATA, инструкции за обработка и коментари

- **FLWR** (произнася се както "flower")

- Изразяване за итерация и за свързване на променливи до междинни резултати
- Полезни са за изчисляване на свързването между два или повече документа и за реструктуриране на данните.
- FLWR означава ключовите думи **FOR, LET, WHERE** и **RETURN** – четири възможни клаузи

# Основни форми на XQuery изрази 4/5

- Сортиращи изрази

- Предоставят начин да се контролира реда на елементите в последователности

- Условни изрази

- Въз основа на ключовите думи IF, THEN и ELSE.

- Количествени изрази (Quantified expressions)

- Поддържат количествено определяне
- Стойността на един количествен израз винаги е верна или неверна

# Основни форми на XQuery изрази 5/5

- Типове данни

- Проверка и манипулиране на типа по време на изпълнение

- Validate

- Израз от тип `validate` валидира своя аргумент по отношение на дефинициите в обхвата от схемата, като използва процеса на валидация, описан в XML Schema

# Примерен XML документ: bib.xml

<bib>

  <book year="1994">

    <title>TCP/IP Illustrated</title>

    <author><last>Stevens</last><first>W.</first></author>

    <publisher>Addison-Wesley</publisher>

    <price> 65.95</price>

  </book>

  <book year="1992">

    <title>Advanced Programming in the Unix environment</title>

    <author><last>Stevens</last><first>W.</first></author>

    <publisher>Addison-Wesley</publisher>

    <price>65.95</price>

  </book>

</bib>

# XQuery пример 1

- Find all books with a price of \$39.95

## XQuery:

```
document("bib.xml")/bib/book[price = 39.95]
```

## Result:

```
<book year="2000">
 <title>Data on the Web</title>
 <author><last>Abiteboul</last><first>Serge</first></author>
 <author><last>Buneman</last><first>Peter</first></author>
 <author><last>Suciu</last><first>Dan</first></author>
 <publisher>Morgan Kaufmann Publishers</publisher>
 <price> 39.95</price>
</book>
```

*Източник: Craig Knoblock. Xquery Tutorial*

# XQuery пример 2

- Find the title of all books published before 1995

## XQuery:

```
document("bib.xml")/bib/book[@year < 1995]/title
```

## Result:

```
<title>TCP/IP Illustrated</title>
```

```
<title>Advanced Programming in the Unix environment</title>
```

# XQuery пример 3 (For Loop)

- List books published by Addison-Wesley after 1991, including their year and title.

## XQuery:

```
<bib>
{
 for $b in document("bib.xml")/bib/book
 where $b/publisher = "Addison-Wesley" and $b/@year >
 1991
 return
 <book year="{ $b/@year }">
 { $b/title }
 </book>
}
</bib>
```

# XQuery пример 3 (For Loop)

- List books published by Addison-Wesley after 1991, including their year and title...

## Result:

```
<bib>
 <book year="1994">
 <title>TCP/IP Illustrated</title>
 </book>
 <book year="1992">
 <title>Advanced Programming in the Unix environment</title>
 </book>
</bib>
```



# XQuery пример 4 (Join)

- For each book found at both bn.com and amazon.com, list the title of the book and its price from each source.

## XQuery:

```
<books-with-prices>
{
 for $b in document("bib.xml")//book,
 $a in document("reviews.xml")//entry
 where $b/title = $a/title
 return
 <book-with-prices>
 { $b/title }
 <price-amazon>{ $a/price }</price-amazon>
 <price-bn>{ $b/price }</price-bn>
 </book-with-prices>
}
</books-with-prices>
```

# XQuery пример 4 (Join)

- For each book found at both bn.com and amazon.com, list the title of the book and its price from each source.

## Result:

```
<books-with-prices>
 <book-with-prices>
 <title>TCP/IP Illustrated</title>
 <price-amazon>65.95</price-amazon>
 <price-bn>65.95</price-bn>
 </book-with-prices><book-with-prices>
 <title>Advanced Programming in the Unix
environment</title>
 <price-amazon>65.95</price-amazon>
 <price-bn>65.95</price-bn>
 </book-with-prices><book-with-prices>
 <title>Data on the Web</title>
 <price-amazon>34.95</price-amazon>
 <price-bn>39.95</price-bn>
 </book-with-prices>
</books-with-prices>
```

# XQuery пример 5

## (Grouping + quantifier)

- For each author in the bibliography, list the author's name and the titles of all books by that author, grouped inside a "result" element.

### XQuery:

```
<results>
{
 for $a in distinct-values(document("bib.com")//author)
 return <result>
 { $a }
 {
 for $b in document("http://bib.com")/bib/book
 where some $ba in $b/author satisfies deep-equal($ba,$a)
 return $b/title
 }
 </result>
}
</results>
```

distinct-values function returns a sequence of unique atomic values from \$arg.

deep-equal returns true if the \$par1 and \$par2 sequences contain the same values, in the same order

# XQuery пример 5

## (Grouping + quantifier)

- For each author in the bibliography, list the author's name and the titles of all books by that author, grouped inside a "result" element.

### Result:

```
<results>
 <result>
 <author>
 <last>Stevens</last>
 <first>W.</first>
 </author>
 <title>TCP/IP Illustrated</title>
 <title>Advanced Programming in the Unix environment</title>
 </result>
 <result>
 <author>
 <last>Abiteboul</last>
 <first>Serge</first>
 </author>
 <title>Data on the Web</title>
 </result>

</results>
```

# XQuery пример 6 (Sorting)

- List the titles and years of all books published by Addison-Wesley after 1991, in alphabetic order.

## XQuery:

```
<bib>
{
 for $b in document("www.bn.com/bib.xml")//book
 where $b/publisher = "Addison-Wesley" and $b/@year > 1991
 return
 <book>
 { $b/@year }
 { $b/title }
 </book>
 sortby (title)
}
</bib>
```

# XQuery пример 6 (Sorting)

- List the titles and years of all books published by Addison-Wesley after 1991, in alphabetic order.

## Result:

```
<bib>
 <book year="1992">
 <title>Advanced Programming in the Unix
environment</title>
 </book>
 <book year="1994">
 <title>TCP/IP Illustrated</title>
 </book>
</bib>
```

# XQuery пример 7 (Recursion)

- Convert a sample document from "**partlist**" format to "**parttree**" format. In the result document, part containment is represented by containment of one `<part>` element inside another. Each part that is not part of any other part should appear as a separate top-level element in the output document.

- partlist.xml:**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<partlist>
```

```
 <part partid="0" name="car"/>
```

```
 <part partid="1" partof="0" name="engine"/>
```

```
 <part partid="2" partof="0" name="door"/>
```

```
 <part partid="3" partof="1" name="piston"/>
```

```
 <part partid="4" partof="2" name="window"/>
```

```
 <part partid="5" partof="2" name="lock"/>
```

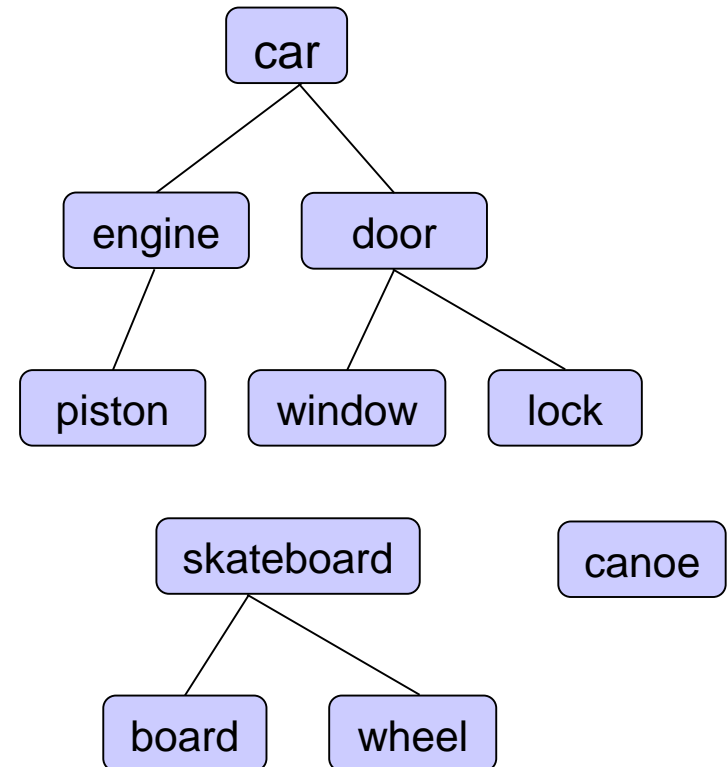
```
 <part partid="10" name="skateboard"/>
```

```
 <part partid="11" partof="10" name="board"/>
```

```
 <part partid="12" partof="10" name="wheel"/>
```

```
 <part partid="20" name="canoe"/>
```

```
</partlist>
```



# XQuery пример 7 (Recursion)

- Convert the sample document from "partlist" format to "parttree" format.

## Result:

```
<parttree>
 <part partid="0" name="car">
 <part partid="1" name="engine">
 <part partid="3" name="piston"/>
 </part>
 <part partid="2" name="door">
 <part partid="4" name="window"/>
 <part partid="5" name="lock"/>
 </part>
 </part>
 <part partid="10" name="skateboard">
 <part partid="11" name="board"/>
 <part partid="12" name="wheel"/>
 </part>
 <part partid="20" name="canoe"/>
</parttree>
```



# XQuery пример 7 (Recursion)

- Convert a sample document from "partlist" format to "parttree" format.

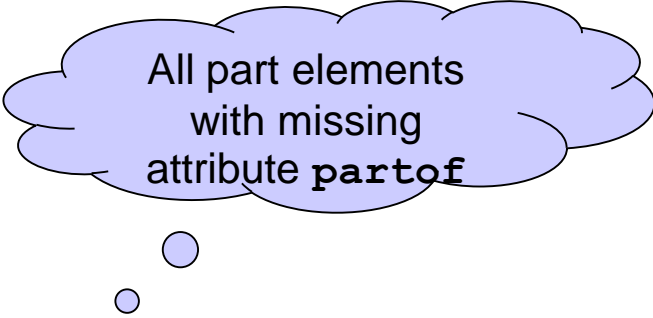
## XQuery:

```
define function one_level (element $p) returns element
{
 <part partid="{ $p/@partid }" name="{ $p/@name }" >
 {
 for $s in document("partlist.xml")//part
 where $s/@partof = $p/@partid
 return one_level($s)
 }
 </part>
}

</parttree>

{
 for $p in document("partlist.xml")//part[empty(@partof)]
 return one_level($p)
}

</parttree>
```



All part elements  
with missing  
attribute **partof**

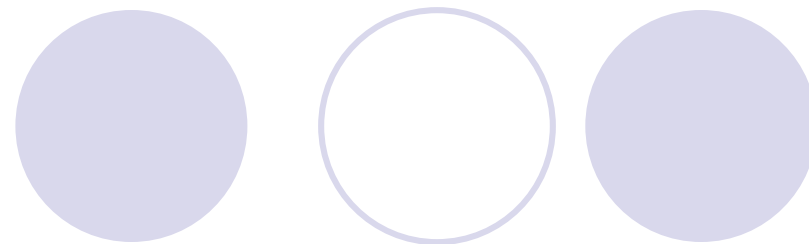
# XQuery поддръжка за RDBMS

- Oracle XQuery Engine
  - <http://www.oracle.com/technology/tech/xml/xquery/index.html>
- Introduction to XQuery in SQL Server
  - [http://msdn.microsoft.com/en-us/library/ms345122\(SQL.90\).aspx](http://msdn.microsoft.com/en-us/library/ms345122(SQL.90).aspx)
- Query DB2 XML data with XQuery
  - <http://www.ibm.com/developerworks/data/library/techarticle/dm-0604saracco/>
- DataDirect: Data Integration Suite – MySQL Database Support
  - <http://www.datadirect.com/products/data-integration/datasources/databases/mysql/index.ssp>

# Заключение 1/2

- Основната употреба на XPath е за определяне на части от документа, които да се обработват, но също така XPath изрази могат да се използват и за изчисления или обработка на низове, за проверка на логически условия и др.
- Механизмът, дефиниран от XPath спецификациите е ефективен и гъвкав, но до известна степен е и ограничен. Така например, XPath не позволява сложни търсения с обединения и друга комплексна обработка на XML документи, която може да се реализира процедурно например с използване на DOM.
- Трябва да се подчертае, че изчисляването на XPath израза връща множество от възли, а не XML документ. Ето защо XPath не се използва самостоятелно, а винаги като спомагателно средство за адресиране или сравнение на секции от XML документа.

# Заключение 2/2



- XQuery е проста алтернатива на XSLT, JSP, ASP, Servlet, CGI, PHP, ....
- Програмите на XQuery могат да изпълнят повечето задачи, традиционно решавани с посочените по-горе инструменти, но все пак са много по-лесни за учене и по-лесни за писане.
- Възможно е да се разшири XQuery за UPDATE и INSERT в XML база данни
- Все още липсва достатъчна подкрепа за XQuery от страна на индустрията

