XML

Martin Klíma

XML – co to je?

XML

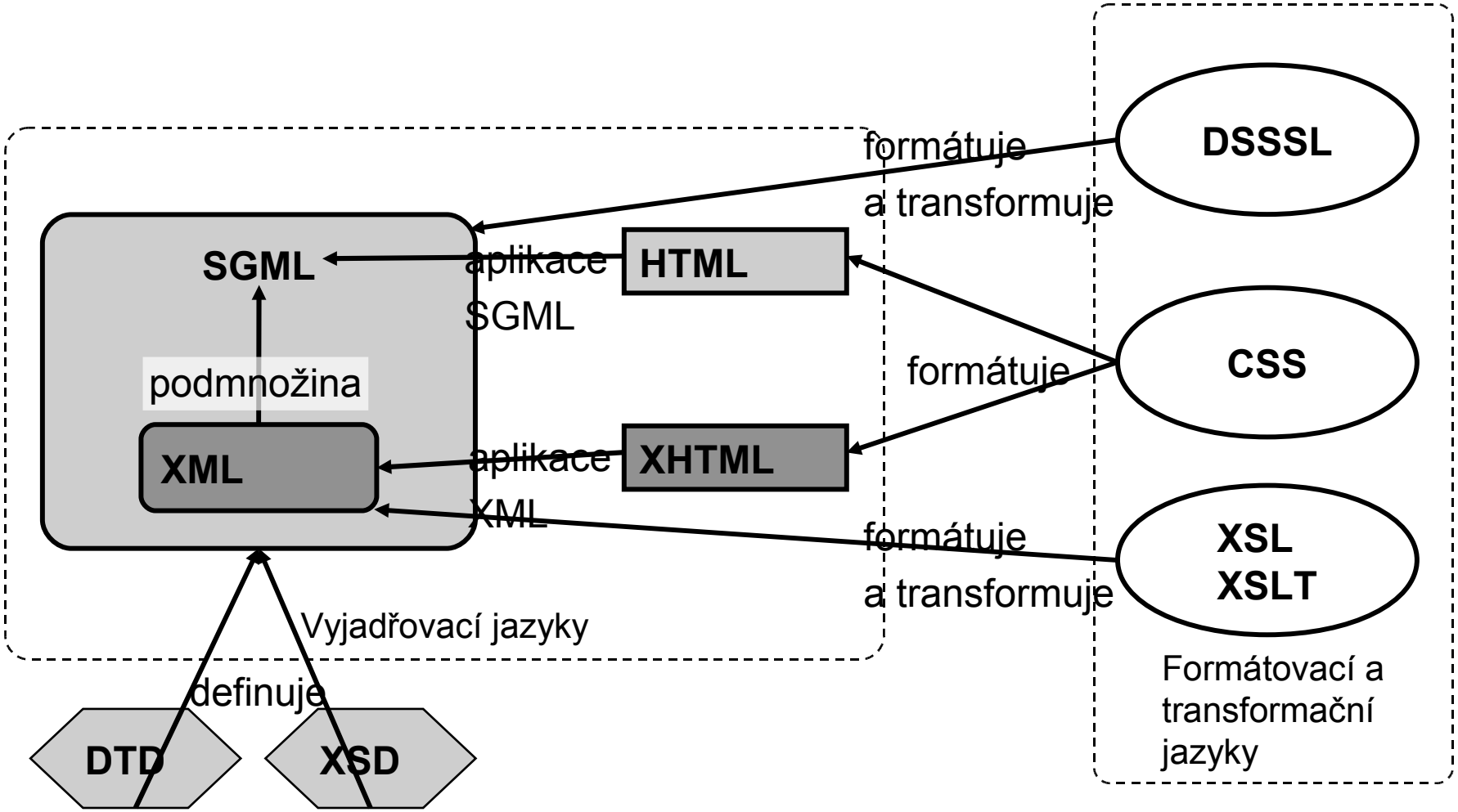
eXtensible Markup Language,

česky *rozšiřitelný značkovací jazyk*

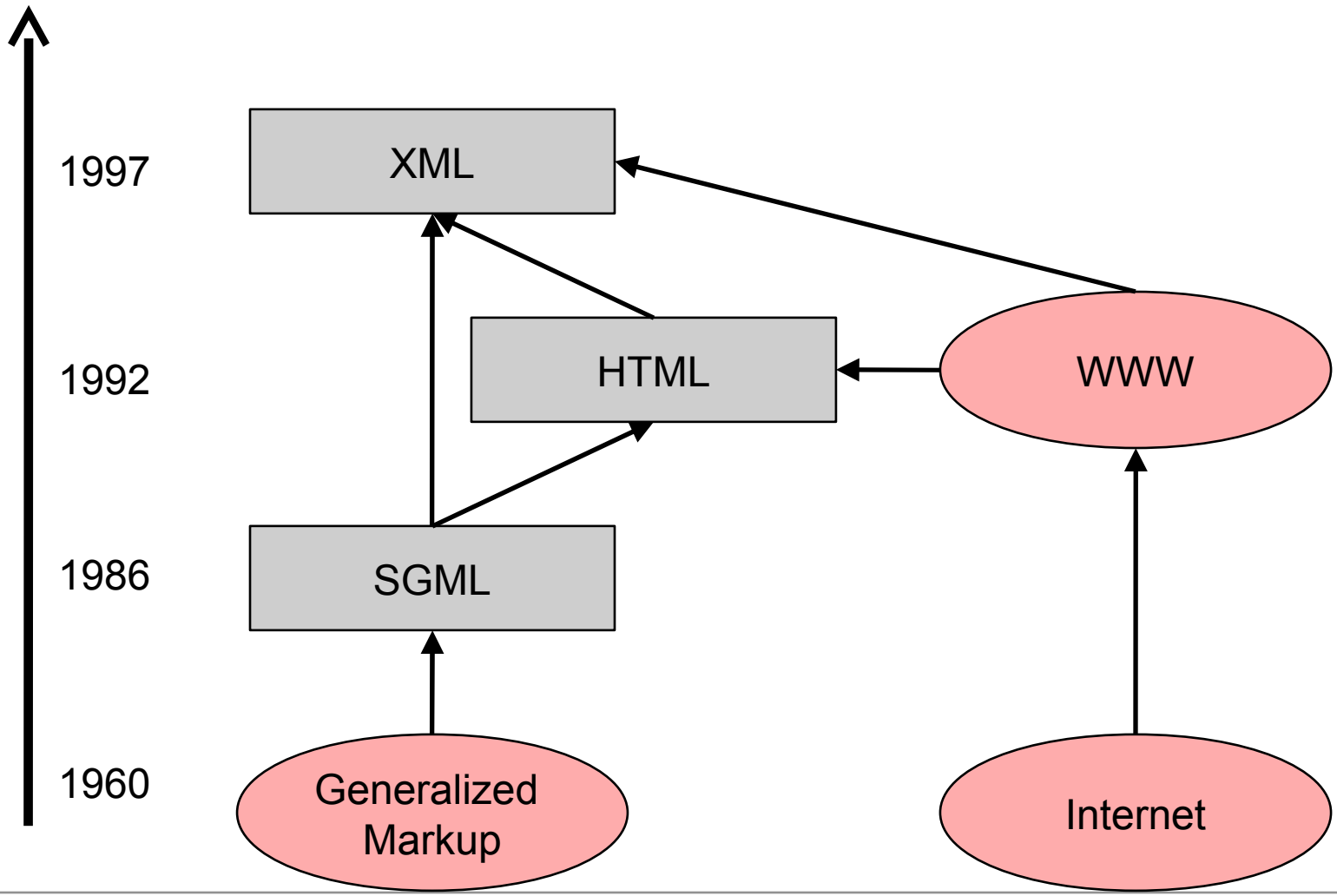
obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C.

Umožňuje snadné vytváření konkrétních značkovací jazyků pro různé účely a široké spektrum různých typů dat.

HTML a jeho vztah k ostatním jazykům

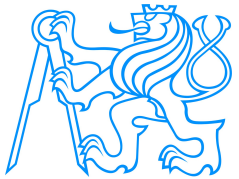


Historie XML



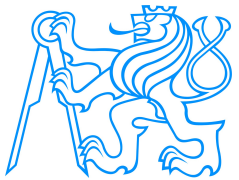
Související technologie

- 1960 GML *(General Markup Language)* vyvinut v IBM pro přenos dokumentů mezi různými platformami
- 1986 SGML *(Standard General Markup Language)* přijato jako ISO standard. Umí reprezentovat téměř všechny dokumenty, značně složitý
- 1992 HTML *(Hypertext Markup Language)* vyvinuto v CERNu, je to aplikace SGML (definováno pomocí DTD)
- 1997 XML *(eXtensible Markup Language)* zjednodušení SGML pro praktické použití konzorciem W3C



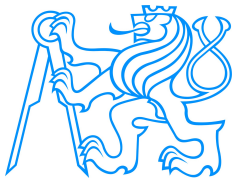
XML - vlastnosti

- Platformově nezávislý
  - textový dokument s pevně definovanými vlastnostmi
- Mezinárodní podpora
  - staví na Unicode, může tedy obsahovat všechny znaky všech národních abeced
  - XML dokument může obsahovat více různých jazyků najednou
- Tagy mají většinou názvy přímo vyjadřující jejich význam
  - XML dokumenty jsou proto dobře čitelné i pouhým okem
- Široká podpora většiny moderních jazyků
  - parsery, kontrola správnosti podle gramatiky



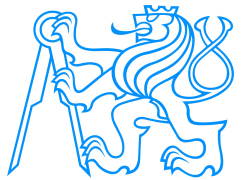
XML - vlastnosti 2

- XML používá hyperlinky
  - XLink, XPointer, XPath
- XSL (XML Stylesheet Language) je jazyk, který definuje, jak se má daný XML dokument transformovat do jiné podoby. Výsledná podoba nemusí být XML.
- CSS (Cascading Style Sheets) jazyk pro formátování dokumetů.
  - CSS se stará o vzhled, zatímco informační obsah a struktura dokumentu je uložena v XML
  - CSS není XML formát
- Automatická kontrola správnosti dokumentů
  - jazyk pro definici struktrury dokumentu DTD



Použití XML

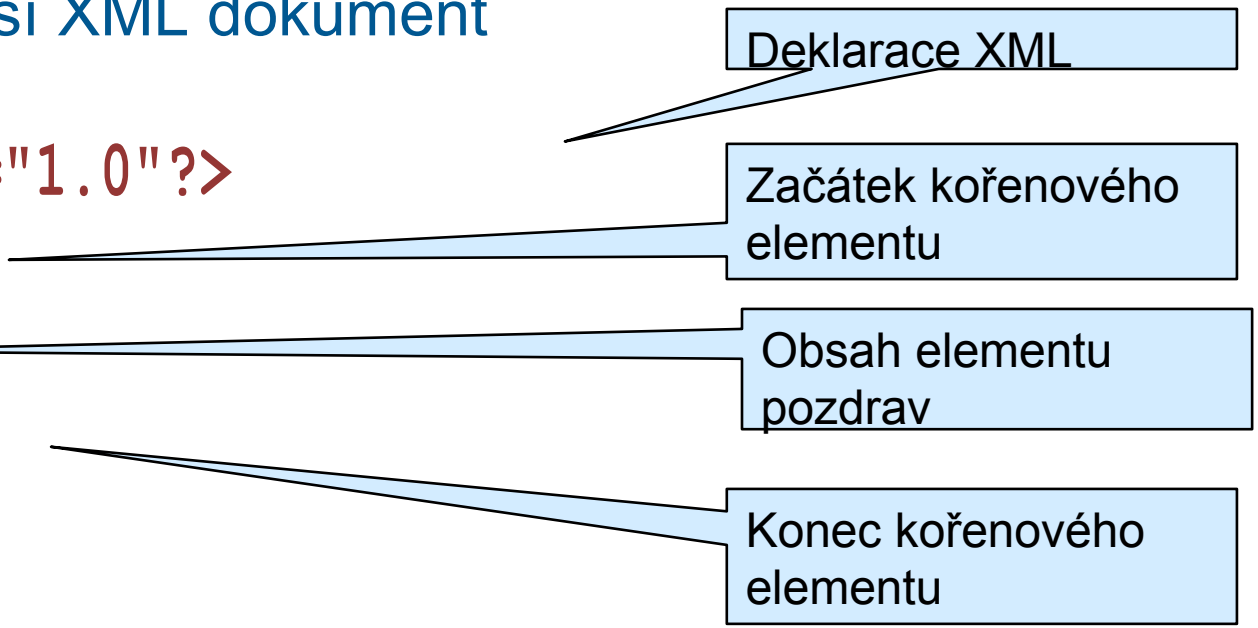
- Dokumenty
  - docx, pptx, xlsx, …
  - XHTML
  - Konverze dokumentů
- B2B (Business to Business)
  - Výměna dat mezi obchodními partnery
  - Požadavek platformové nezávislosti
- Import dat, export dat, viz také B2B
- Webové technologie
- Konfigurační soubory
- …



Základy jazyka XML

- Nejjednodušší XML dokument

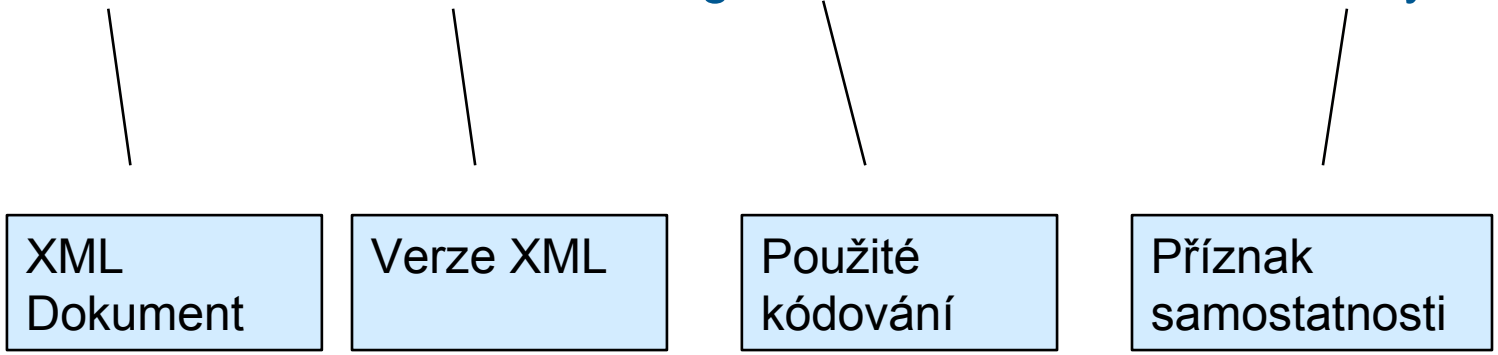
```
<?xml version="1.0"?>
<pozdrav>
Ahoj
</pozdrav>
```



Části XML dokumentu

- Deklarace XML

```
<?xml version="1.0" encoding="ISO-8859-2" standalone="yes"?>
```



- Deklarace je povinná

Části XML dokumentu

Každý XML dokument MUSÍ mít právě jeden kořenový element

- nesmí tedy mít žádný nebo více než jeden

```
<?xml version="1.0"?>
<pozdrav>
Ahoj
</pozdrav>
```

Kořenový element

```
<pozdrav>
Ahoj podruhé
</pozdrav>
```

Neplatné XML, 2 kořenové elementy

Elementy

```
<?xml version="1.0"?>
<dokument>
   <paragraf>
   text text text
    blablabla
   </paragraf>
   <oddelovac />
   <paragraf>
   další text
   blablabla
   </paragraf>
</dokument>
```

Začátek (kořenového) elementu **dokument**

Začátek elementu **paragraf**

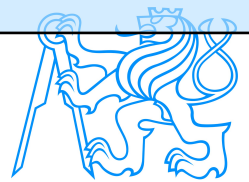
Konec elementu **paragraf**

Prázdný element **oddelovac**

Začátek elementu **paragraf**

Konec elementu **paragraf**

Konec (kořenového) **elementu** dokument



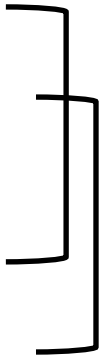
Elementy

- Elementy musí být ukončeny

  <body> </body>
- Pokud je element prázdný, je hned ukončen

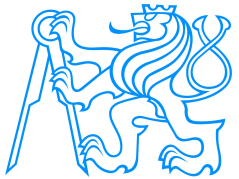
- Elementy se mohou vnořovat ale ne křížit

```
<?xml version="1.0"?>
<dokument>
   <strong>
      <paragraf>
      text text text
      </strong>
   </paragraf>
</dokument>
```



! křížení tagů

Computer Graphics Group



Komentáře

Komentáře mají tento tvar:

```
<!--  zde je text
   komentráře -->
```

Komentáře musí následovat až za XML deklarací

Komentáře nesmí být uvnitř tagů

```
<dokument <!--
komentář 1 --> >
</dokument>
```

```
<?xml version="1.0"?>
<!-- komentář 1 -->
<dokument>
   <paragraf>
   text text text
    blablabla
   </paragraf>
   <oddelovac />
   <paragraf>
   <!-- komentář 2 -->
   další text
   blablabla
   </paragraf>
</dokument>
```

Znaky

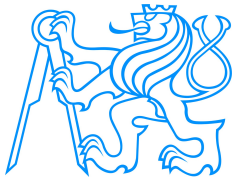
- Data uložená v XML souborech je nutné kódovat

&amp;	&
&lt;	<
&gt;	>
&quot;	“
&apos;	‘

Znaky II

- Při vkládání velkých objemů dat může být kódování nepohodlné
- Řešení: CDATA

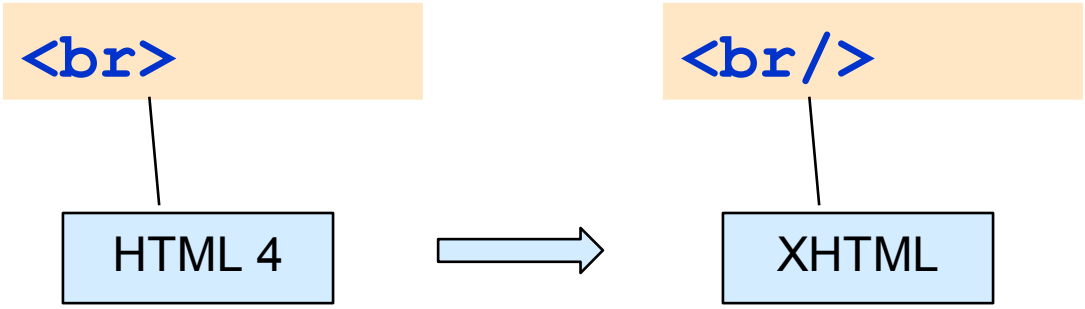
```
<dokument>
<![CDATA[
tady si můžu dělat co chci a je to ok < blabla>
]]>
</dokument>
```

Tagy

- Tagy = značky mají jméno
- Jméno musí začínat písmenem nebo znakem _
- Jsou case-sensitive!

- Všechny tagy musí být uzavřené
- Pokud tag není párový, musí se uzavřít sám v sobě



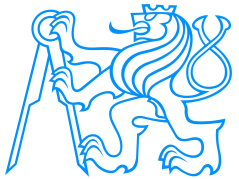
Atributy

- Tagy mohou mít atributy
- Tag může mít 0 a více atributů

```
<?xml version="1.0"?>
<dokument>
   <paragraf jazyk="CZ" zarovnani="vlevo">
   text text text
   </paragraf>
   <oddelovac />
   <paragraf jazyk="EN">
   more text blablabla
   </paragraf>
</dokument>
```

Atributy II

- Pravidla pro jména atributů jsou stejná jako pro jména tagů
- Hodnoty atributů jsou uzavřeny v “ nebo v ‘
- Hodnoty nemají typ, jsou to prostě řetězce



Well – formed dokumenty

XML dokument je well – formed, pokud:

- Má na začátku XML deklaraci
- Má právě jeden kořenový element
- Elementy obsahující data mají začáteční i koncový tag
- Nepárové elementy jsou ukončeny **/>**
- Elementy se nesmí křížit
- Hodnoty atributů musí být uzavřeny v “ nebo ‘
- Znaky < a & mohou být použity jenom jako významové znaky
- Nevýznamové znaky musí být zakódovány

Formátování - CSS

- XML definuje obsah, ne formátování
- Formátování je obsaženo v jiné struktuře: CSS

```
<?xml version="1.0"?>
<?xml-stylesheet type=”text/css” href=”formatovani.css”?>
<dokument>
   <paragraf>
   text text text
   </paragraf>
   <oddelovac />
   <paragraf>
   další text blablabla
   </paragraf>
</dokument>
```


viz soubor.xml,
formatovani.css

Zobrazení

Prohlížeč neví, jak dokument zobrazit

```
<?xml version="1.0" ?>
- <dokument>
    <paragraf>text text text</paragraf>
    <oddelovac />
    <paragraf>dalsi text blablabla</paragraf>
  </dokument>
```

S CSS

text text text
dalsi text blablabla

```
paragraf {
   font-size: 3em;
   display: block;
   background-color:
yellow;
}
```

viz soubor.xml, formatovani.css

Výsledek vykreslení XML dokumentu

Pravidla pro zobrazení v souboru formatovani.css

Definice struktury dokumentu - gramatika

Definuje se pomocí pravidel napsaných v DTD souboru

DTD = Document Type Declaration

DTD specifikuje gramatiku XML souboru

Parsery umí kontrolovat proti této gramatice

XML je well formated pokud neporušuje základní pravidla formátování

XML je valid pokud splňuje pravidla příslušné gramatiky

DTD ELEMENT

ELEMENT definuje strukturu a pořadí, počty elementů v dokumetu

obecná definice

<!ELEMENT jméno_elementu (obsah_elementu)>

př.:

<!ELEMENT paragraf (#PCDATA)>

Element paragraf obsahuje volný text

př.:

<!ELEMENT dokument (paragraf+, oddelovac*)+>

Element dokument obsahuje jeden nebo více paragrafů a 0 nebo více oddělovačů

DTD

Tato direktiva říká, že definice struktury tohoto dokumentu je v souboru dokument.dtd

XML

```
<?xml version="1.0"?>
<!DOCTYPE dokument
SYSTEM "dokument.dtd">
<dokument>
    <paragraf>
    text text text
    </paragraf>
    <oddelovac />
    <paragraf>
    blablabla
    </paragraf>
</dokument>
```

DTD (soubor dokument.dtd)

```
<!ELEMENT dokument
   (paragraf+, oddelovac*)+>

<!ELEMENT paragraf (#PCDATA)>

<!ELEMENT oddelovac EMPTY>
```

paragraf obsahuje volný text, žádný element

oddělovač neobsahuje nic, je prázdný (empty)

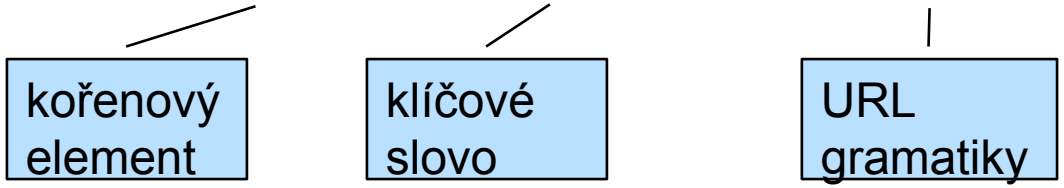
viz
dokument_dtd1.xml
dokument.dtd

Doctype – externí definice DTD

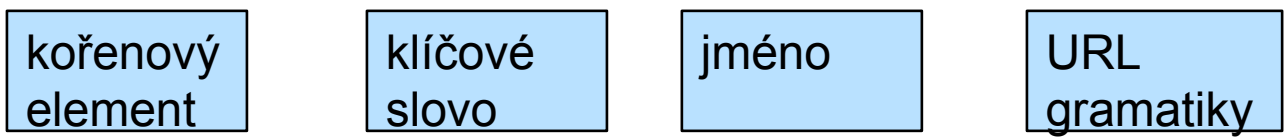
- DTD může být referována jako externí zdroj
- K určení umístění se používá URL
  - může být relativní či absolutní
  - SYSTEM - pro použití jedním autorem nebo lokálním kolektivem
  - PUBLIC – pro veřejné použití, definuje navíc ještě jnéno pro DTD
- V definici je jméno kořenového (root) elementu

DTD

```
<?xml version="1.0"?>
<!DOCTYPE dokument SYSTEM "dokument.dtd">
```




```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```



DTD – další možnosti

Element může obsahovat buď jeden, druhý nebo třetí element

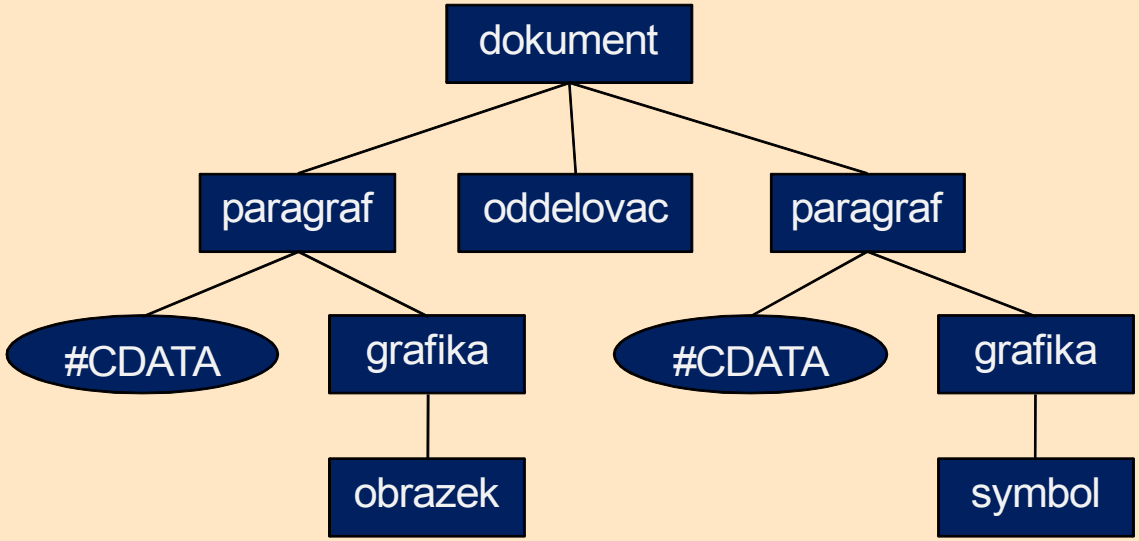
Př. element **`grafika`** obsahuje jeden z těchto elementů:
`obrazek` nebo **`symbol`** nebo **`animace`**

V DTD to vyjádříme takto:

```
<!ELEMENT grafika (obrazek|symbol|animace)>
<!ELEMENT obrazek EMPTY>
<!ELEMENT symbol EMPTY>
<!ELEMENT animace EMPTY>
```

DTD – další možnosti pokačování XML dokument

```
<?xml version="1.0"?>
<!DOCTYPE dokument SYSTEM "dokument_graficky.dtd">
<dokument>
    <paragraf>
       text text text
       <grafika>
          <obrazek/>
       </grafika>
    </paragraf>
    <oddelovac/>
    <paragraf>
       další text blablabla
       <grafika>
       <symbol/>
       </grafika>
    </paragraf>
</dokument>
```



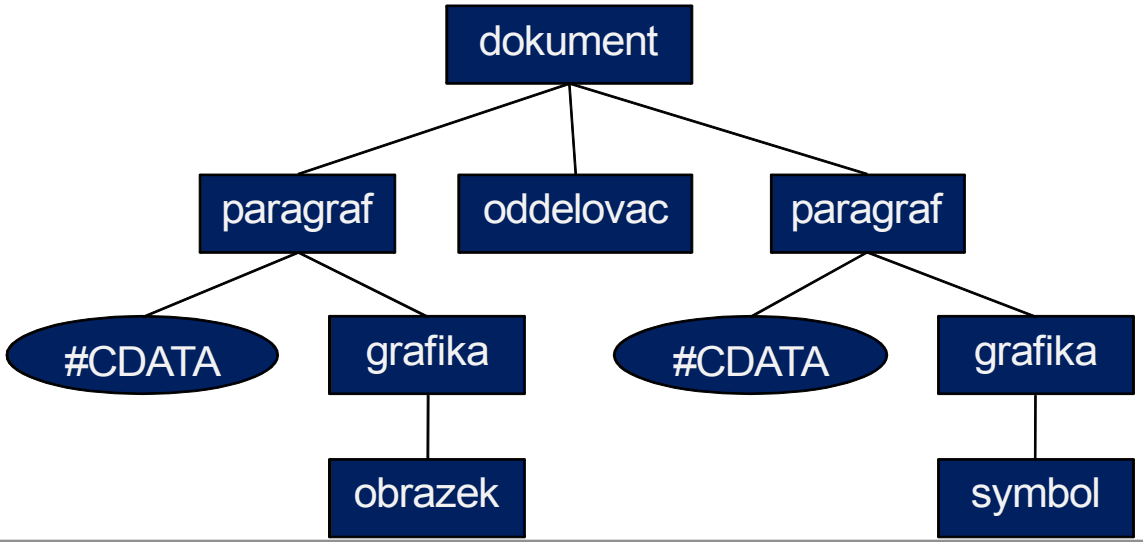
viz
graficke_elementy.xml

DTD – další možnosti pokačování DTD

```
<!ELEMENT dokument (paragraf+, oddelovac*)+>
<!ELEMENT paragraf (#PCDATA|grafika)*>
<!ELEMENT oddelovac EMPTY>
<!ELEMENT grafika (obrazek | symbol | animace)>
<!ELEMENT obrazek EMPTY>
<!ELEMENT symbol EMPTY>
<!ELEMENT animace EMPTY>
```

Mix #PCDATA a Element

Možnosti



viz dokument_graficky.dtd

DTD a entity

- Entita je nějaká jednotka, která má svůj identifikátor v XML dokumentu a je možné jí rozvinout

Př.:

```
<!ENTITY CVUT "České vysoké učení technické">
```

jméno

rozvinutí

```
<?xml version="1.0"?>
<!DOCTYPE dokument SYSTEM "entity.dtd">
<dokument>
        <paragraf>
                text text text &CVUT;
                <grafika>
                        <obrazek/>
                </grafika>
        </paragraf>
</dokument>
```

použití entity

viz
entity.xml
entity.dtd

DTD – předdefinované entity

&amp;	&
&lt;	<
&gt;	>
&quot;	“
&apos;	‘

```
<!ENTITY lt "&#38;#60;">
<!ENTITY gt "&#62;">
<!ENTITY amp "&#38;#38;">
<!ENTITY apos "&#39;">
<!ENTITY quot "&#34;">
```

DTD – externí entity

Za entity můžeme prohlásit celý XML soubor.

Př.: (soubor podpis.xml)

```
<?xml version="1.0"?>
<podpis>
    <copyright>Martin Klima</copyright>
</podpis>
```

V DTD pak můžeme tento soubor nazvat entitou

```
<!ENTITY SIG SYSTEM "podpis.xml">
```

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE dokument SYSTEM "entity_podpis.dtd">
<dokument>
        <paragraf>
                text text text &SIG;
                <grafika>
                        <obrazek/>
                </grafika>
        </paragraf>
</dokument>
```

viz
podpis.xml

DTD a atributy

Předpokládejme, že chceme odstavci přiřadit nějaké atributy a jejich možné hodnoty. Zeleně je označena **default** hodnota. Řekněme, že atribut zarovnání je **povinný**.

- zarovnání: **vlevo**, vpravo, doprostred
- id: identifikátor obsahující jakýkoli řetězec
- radkovani: **1**, 2, 3, ….
- odsazeni: jakékoli číslo, nepovinný atribut

```
<?xml version="1.0"?>
<!DOCTYPE dokument SYSTEM "atributy.dtd">
<dokument>
      <paragraf zarovnani="vlevo" id="p1"
radkovani="1">
            text text text
      </paragraf>
</dokument>
```

viz atributy1.xml

Atributy musí být zapsány v DTD

Zápis v DTD

```
...
...
...
<!ATTLIST paragraf
    zarovnani CDATA "vlevo"
    id        CDATA  #REQUIRED
    radkovani CDATA "1"
    odsazeni  CDATA #IMPLIED
  >
```

Atributy elementu **paragraf**

Atribut zarovnani má default hodnotu "vlevo"

Atribut id je povinný

Atribut odsazeni je nepovinný a nemá default hodnotu

viz atributy.dtd

DTD – typy atributů

CDATA	jakýkoli text
Výčet	výčet možných hodnot
ID	jednoznačný identifikátor v rámci dokumentu
IDREF	hodnota ID atributu nějakého elementu
NMTOKEN	XML jméno (NameChar)+
NMTOKENS	více XML jmen oddělených čárkou
IDREFS	více IDček elementů oddělených čárkami
ENTITY	jméno entity deklarovné v DTD
ENTITIES	jména více entit oddělená čárkami
NOTATION	jméno notace deklarované v DTD

namechar::= Letter | Digit | '.' | '-' | '_' | ':' | CombiningChar | Extender

DTD Atributy - výčet

Zpět k požadavku na atribut **zarovnani**

zarovnání: **vlevo**, vpravo, doprostred

Zde nechceme, aby byl přípustný jiný atribut než vyjmenovaný a default je vlevo

```
...
...
<!ATTLIST paragraf
    zarovnani (vlevo | vpravo | doprostred) "vlevo"
    id         CDATA  #REQUIRED
    radkovani CDATA "1"
    odsazeni   CDATA #IMPLIED
  >
```

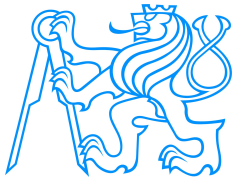
Výčet možných hodnot atributu zarovnani

Default hodnota atributu zarovnani

viz atributy_lepsi.dtd

Jmenné prostory (namespaces)

- Umožňují používat několik druhů značek v jednom dokumentu
- Značky mohou mít stejná jména, ale díky namespace je dokážeme rozlišit



Namespace - příklad

- Dvě různé tabulky
- Každá vychází z jiné definice
- Liší se tedy strukturou

```
<table>
    <tr>
        <td>Apples</td>
        <td>Bananas</td>
    </tr>
</table>
```

```
<table>
    <name>African Coffee Table</name>
    <width>80</width>
    <length>120</length>
</table>
```

Mohu tyto tabulky dostat do jednoho dokumetu?

- 2 možnosti
  - použít prefix
  - použít namespace

```
<h:table>
    <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
    </h:tr>
</h:table>
```

```
<f:table>
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
</f:table>
```

Namespaces - zápis

- pomocí atributu xmlns:*prefix*
- hodnota je URI jmenného prostoru

Prefix xml a xmlns jsou rezervované

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">
   <h:tr>
   <h:td>Apples</h:td>
   <h:td>Bananas</h:td>
   </h:tr>
</h:table>
```

```
<f:table xmlns:f="http://www.w3schools.com/furniture">
   <f:name>African Coffee Table</f:name>
   <f:width>80</f:width>
   <f:length>120</f:length>
</f:table>
```

Default namespace

- Pokud nadřazenému elementu řeknu, v jakém je jmenném prosturu, jeho potomci jsou v něm také

```
<table xmlns="http://www.w3.org/TR/html4/">
   <tr>
       <td>Apples</td>
       <td>Bananas</td>
   </tr>
</table>
```

```
<table xmlns="http://www.w3schools.com/furniture">
   <name>African Coffee Table</name>
   <width>80</width>
   <length>120</length>
</table>
```

Namespace – definice na začátku souboru

```
<?xml version=”1.0”?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/XSL/Transform/1.0"
       xmlns:html="http://www.w3.org/TR/REC-html40">
  <xsl:template match="PERIODIC_TABLE">
     <html:html>
        <xsl:apply-templates/>
     </html:html>
  </xsl:template>
  <xsl:template match="ATOM">
     <html:p>
       <xsl:apply-templates/>
     </html:p>
  </xsl:template>
</xsl:stylesheet>
```

Definuje použité namespace

Reference

http://www.xml.com/

http://www.biztalk.org/

http://www.xml.org/

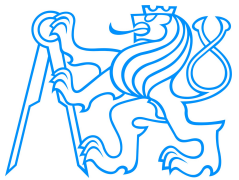
http://www.oasis-open.org/cover/

http://zvon.vscht.cz/

http://www.xmlsoftware.com/

http://www.w3.org/XML/

http://www.wapserver.cz/



Technologie založené na XML

XPath

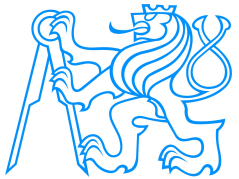
XPath

- Je to jazyk pro hledání informace v XML dokumentu
- Založen na XML, tj. splňuje XML pravidla a má DTD
- Na XPath staví další jazyky jako XPointer a XQuery

- XPointer je jazyk pro provázání dokumentů, umožňuje ukázat na konkrétní část nějakého XML dokumetu
- XQuery je jazyk pro dotazování nad XML daty. Je to obdoba SQL nad relačními databázemi

XPath

- Je to syntaxe pro pojmenovávání částí XML dokumetů
- Vytváří výrazy pro navigaci v XML dokumetu
- Obsahuje sadu funkcí (více než 100)
- Je důležitou součástí XSLT
- Je to W3C standard



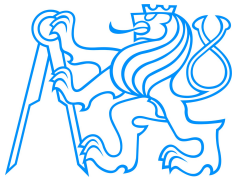
XPath - terminologie

XPath rozeznává tyto XML struktury (nodes)

1. Kořen (root)
2. Elementy
3. Text
4. Atributy
5. Jmenné prostory (namespaces)
6. Instrukce
7. Komentáře

XPath

- XML dokument je strom z uzlů a lze ho procházet
- Co ve stromu není vidět
  - pořadí atributů, jak byly napsány
  - typ uvozovek resp. apostrofů
  - entity a kódy znaků
  - deklarace jmenných prostorů



Co tedy XPath přesně dělá

- Vyhodnocuje daný výraz na XML stromu
- Výsledek je výběr, tj. podmnožina uzlů stromu
- …nebo syntaktická chyba ☺

```
//img[not(@alt)]          vybere všechny obrázky, které nemají
                          atribut alt

count(//img)              vrátí počet obrázků

/descendant::img[3]/@src
                          vrátí hodnotu atributu src třetího
obrázku v pořadí

starts-with(/html/@lang, 'en')
                          otestuje, zda je html dokument v
angličtině
```

Adresování uzlů

Výraz	Význam
jméno_uzlu	vybere všechny děti uzlu s daným jménem
/	výber od kořene
//	vybere uzly v dokumentu od aktuálního uzlu, které splňují podmínku. Nezáleží na tom, jak hluboko jsou.
.	vybere aktuální uzel
..	vybere rodiče
@	vybere atributy

Příklad

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
    <book>
      <title lang="en">Harry Potter</title>
      <author>J K. Rowling</author>
      <year>2005</year>
      <price>29.99</price>
    </book>
</bookstore>
```

Výběr - ukázky

Výraz	Výsledek
bookstore	Vybere všechny děti elementu bookstore
/bookstore	Vybere kořenový element bookstore. Jedná se o absolutní cestu.
bookstore/book	Vybere všechny elementy book, které jsou děti elementu bookstore
//book	Vybere všechny elementy book, kdekoli se nacházejí
bookstore//book	Vybere všechny elementy book, které jsou potomky elementu bookstore, jakkoli hluboko se nacházejí.
//@lang	Vybere všechny atributy se jménem lang.

Predikáty

- Používají se k výběru uzlu s význačnou vlastností

Výraz	Výsledek
/bookstore/book[0]	Vybere první element book, který je první syn elementu bookstore.
/bookstore/book[last()]	Vybere posledního syna téhož.
/bookstore/book[last()-1]	Vybere předposledního syna téhož.
/bookstore/book[position()<3]	Vybere první dva elementy book, které jsou syny elementy bookstore.
//title[@lang]	Všechny elementy title s atributem lang.
//title[@lang='eng']	Stejné a navíc hodnotu eng.
/bookstore/book[price>35.00]	Všechny elementy book, které jsou syny bookstore a mají atribut cena s hodnotu větší než 35.00
/bookstore/book[price>35.00]/title	Všechny elementy title patřící viz předchozí řádek.

Výběr neznámých uzlů

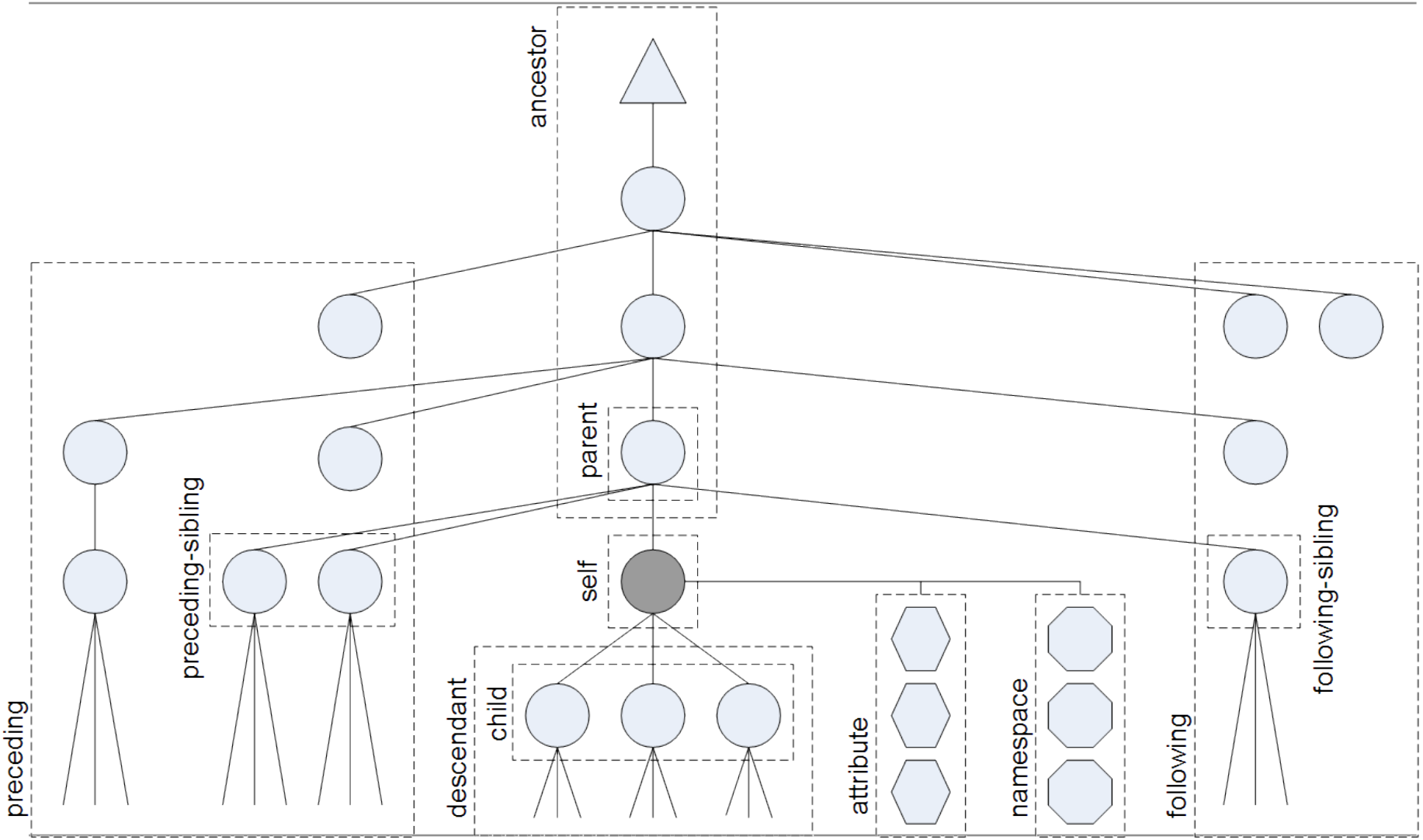
Zástupný znak	Význam
*	Jakýkoli uzel typu element
@*	Jakýkoli uzel typu atribut
node()	Jakýkoli uzel

Výraz	Význam
/bookstore/*	Všechny syny kořenového uzlu bookstore
//*	Všechny elementy v dokumetu
//title[@*]	Všechny elementy title, které mají nějaký atribut

Výběr více uzlů

Výraz	Význam
//book/title | //book/price	Všechny elemety title pod elementem book a všechny el. price pod el. book, kdekoli book je.
//title | //price	Všchny elementy title a price v dokumentu.

Adresování relativně k aktulálnímu uzlu

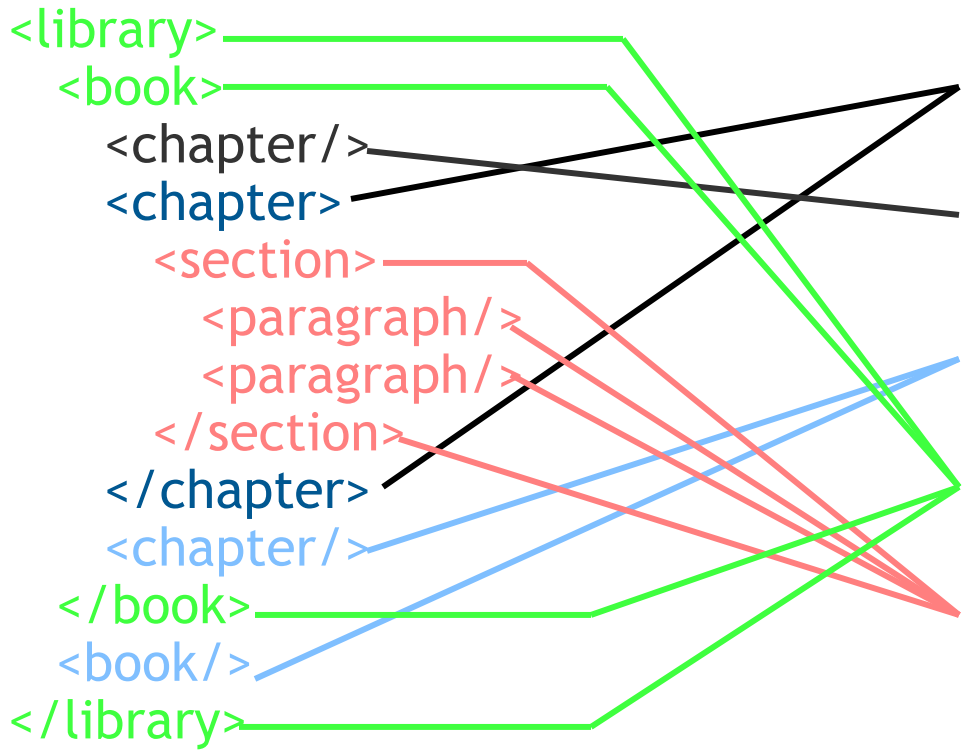


zdroj: http://dret.net/lectures/publishing-spring07/xpath20#(16)

Axes příklady

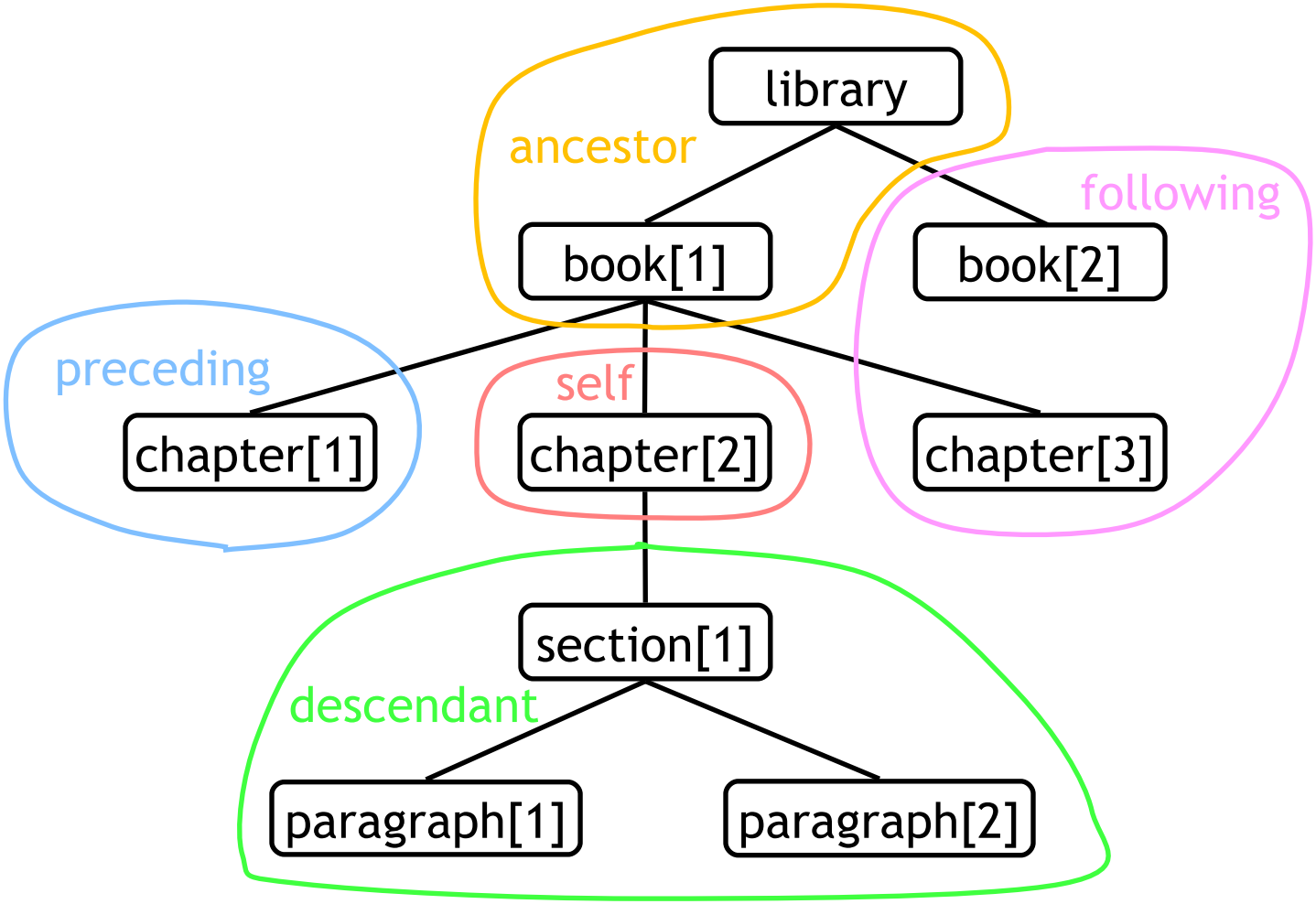
Výraz	Význam
child::book	Všechny book uzly, které jsou synem aktuálního uzlu
attribute::lang	Attribut lang aktuálního uzlu
child::*	Všichni syni aktuálního uzlu
attribute::*	Všechny atributy aktuálního uzlu
child::text()	Všichni syni typu text od aktuálního uzlu
child::node()	Všichni syni aktuálnío uzlu
descendant::book	Všichni potomci book aktuálního uzlu
ancestor::book	Všichni book předchůdci aktuálního uzlu
ancestor-or-self::book	Předchůdci book včetně aktuálního uzlu
child::*/child::price	Všechny uzly price, které jsou vnoučaty aktuálního uzlu

Axes - příklady



- //chapter[2]/self::*
- //chapter[2]/preceding::*
- //chapter[2]/following::*
- //chapter[2]/ancestor::*
- //chapter[2]/descendant::*

Axes DOM


library
ancestor
following
book[1]
book[2]
preceding
self
chapter[1]
chapter[2]
chapter[3]
section[1]
descendant
paragraph[1]
paragraph[2]

Axes zkrácené formy

(nic)	je ekvivalent	child::
@	je ekvivalent	attribute::
.	je ekvivalent	self::node()
.//X	je ekvivalent	self::node()/descendant-or-self::node()/child::X
..	je ekvivalent	parent::node()
../X	je ekvivalent	parent::node()/child::X
//	je ekvivalent	/descendant-or-self::node()/
//X	je ekvivalent	/descendant-or-self::node()/child::X

Operátory

Operátor	Význam	Příklad
|	Dvě sady uzlů	//book | //cd
+	Sčítání	6 + 4
-	Odčítání	6 - 4
*	Násobení	6 * 4
div	Dělení	8 div 4
=	Rovnost	price=9.80
!=	Nerovnost	price!=9.80
<	Menší než	price<9.80
<=	Menší nebo rovno	price<=9.80
>	Větší než	price>9.80
>=	Větší nebo rovno	price>=9.80
or	nebo	price=9.80 or price=9.70
and	a	price>9.00 and price<9.90
mod	zbytek po dělení	5 mod 2

XPath funkce

- Celá řada (kategorií) funkcí
- Mají prefix fn: a URI http://www.w3.org/2005/02/xpath-functions
  - Accessor
  - Error and Trace
  - Numeric
  - String
  - AnyURI
  - Boolean
  - Duration/Date/Time
  - Qname
  - Node
  - Sequence
  - Context

Příklady XPath funkcí

Funkce	Význam
fn:node-name(node)	Jméno uzlu argumentu
fn:abs(*num*)	Absolutní hodnota argumetnu
fn:compare(*comp1,comp2*)	Porovná argumenty jako řetězce
fn:string-length()	Vrací délku řetězce aktuálního uzlu
fn:true()	Vrací hodnotu true
fn:dateTime(*date,time*)	Převede hodnoty na datumčas
fn:root(*node*)	Vrací kořenový uzel
fn:reverse(*(item1,item2,...itemN)*)	Vrací obrácené pořadí itemN,…item2,item1

Příklady XPath funkcí

```
<?xml version="1.0"
encoding="UTF-8"?>
<library>
   <book>
      <chapter/>
      <chapter>
         <section>
            <paragraph/>
            <paragraph/>
         </section>
      </chapter>
      <chapter/>
   </book>
   <book/>
</library>
```

//chapter[count(section)=1]

Vybere chapter, které mají přesně jednoho syna section

//*[name()='section']

//section

Vybere uzel section kdekoli v dokumentu

//*[starts-with(name(), 'sec']

Vybere uzly, jejichž jméno začíná na 'sec' kdekoli v dokumentu

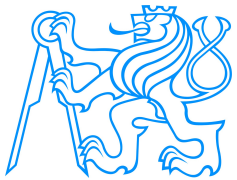
//*[contains(name(), 'ect']

Vybere uzly, jejichž jméno obsahuje řetězec 'ect' kdekoli v dokumentu

XSL Transformace

- Extensible Style Language
  - jazyk pro transformaci a formátování XML dokumentů

- Transformace dokumentů je založena na jeho stromové struktuře
- Definujeme pravidla pro manipulaci s jednotlivými strukturami
- Výstup může být jiný XML dokument nebo jiný typ dokumentu (např. transformace z XHTML do HTML nebo PDF)



XSLT

- Defunuje sadu pravidel
- Transformační pravidlo
  - aplikuje se podle vzoru
  - definuje co se přidá na výstup
  - říká, co se bude dělat dál (např. pokračovat rekurzivně)
- XSLT Processor začne aplikova pravidla od kořenového uzlu
- XSLT používá model dokumentu stejný jako XPath

XSLT – typická struktura

```
<?xml version="1.0" ?>
   <xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

     <!- Tady je transofrmační pravidlo: -->

     <xsl:template match=vzor>
        XSL elementy a výstup
     </xsl:template>

     <!- Další transformační pravidla -->

</xsl:stylesheet>
```

XSLT

- Vzor určuje, na které uzly se bude pravidlo aplikovat
- Obsah těla pravidla, který není v XSL: jmenném prostoru, se přenese rovnou na výstup
- To, co je v XSL: jmenném prostoru se bude dále interpretovat

XSLT příklad

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

<bookstore>
    <!--Authors-->
    <authors>
        <author id="author01">
            <firstname>Josef</firstname>
            <lastname>Novak</lastname>
        </author>
        <author id="author02">
            <firstname>Jana</firstname>
            <lastname>Novotna</lastname>
        </author>
        <author id="author03">
            <firstname>Vladislav</firstname>
            <lastname>Vomacka</lastname>
        </author>
    </authors>
</bookstore>
```

XSLT – příklad cont

```
<?xml version="1.0" ?>
   <xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="html"/>

    <xsl:template match="/">
         <html>
         <head>
             <title>bookstore.xsl</title>
         </head>
         <body>
                  <xsl:apply-templates />
         </body>
         </html>
    </xsl:template>

    <xsl:template match="//author/firstname">
         <div>Jmeno=<xsl:value-of select="." /></div>
    </xsl:template>

     <xsl:template match="//author/lastname">
         <div>Prijmeni=<xsl:value-of select="." /></div>
    </xsl:template>
</xsl:stylesheet>
```

XSLT příklad výsledek

```
<html>
<head>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>bookstore.xsl</title>
</head>
<body>
<div>Jmeno=Josef</div>
<div>Prijmeni=Novak</div>
<div>Jmeno=Jana</div>
<div>Prijmeni=Novotna</div>
<div>Jmeno=Vladislav</div>
<div>Prijmeni=Vomacka</div>
</body>
</html>
```

Provázání XSLT a XML zdroje

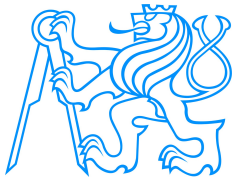
```
<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="bookstore.xsl"?>

<bookstore>
    <!--Authors-->
    <authors>
        <author id="author01">
            <firstname>Josef</firstname>
            <lastname>Novak</lastname>
        </author>
        <author id="author02">
            <firstname>Jana</firstname>
            <lastname>Novotna</lastname>
        </author>
        <author id="author03">
            <firstname>Vladislav</firstname>
            <lastname>Vomacka</lastname>
        </author>
    </authors>
</bookstore>
```

Postup při nalezení a vykonání XSLT

- Vezme se seznam uzlů (na začátku kořen)
- Nalezení včech pravidel, která se na daný uzel aplikují
- Nalezení nejlepšího z nich
- Vykonání nejlepšího z nich

Postup při vykonání pravidla

- Při vykonávání pravidla jsou vybrané uzly považovány za *current.*
- Typicky je vybrána nová množina uzlů ke zpracování
- Rekurze končí, když jsou zpracovány všechny uzly



Řešení konfliktů

- Na daný uzel se může vztahovat více pravidel

```
<xsl:template match="/authors/author">
   <!- prvni pravidlo -->
</xsl:template>

<xsl:template match="//author">
   <!- druhe pravidlo -->
</xsl:template>
```

- Lokální pravdla mají přednost před importovanými
- Pravidla jsou řazena takto od nejnižší priority

  xyz:* foo baz/foo
- Pozor na konkrétní implementaci!

XSL Apply Templates

- Uvnitř pravidla může být výběr nové množiny uzlů
- Tato sada je většinou podmnožina z potomků

  …nemusí ale být (XPath)

```
<xsl:template match="//authors">

   <xsl:apply-template match="author" />

</xsl:template>
```

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

<bookstore>
  <!--Authors-->
  <authors>
     <author id="author01">
        <firstname>Josef</firstname>
        <lastname>Novak</lastname>
     </author>
     <author id="author02">
        <firstname>Jana</firstname>
        <lastname>Novotna</lastname>
     </author>
     <author id="author03">

<firstname>Vladislav</firstname>

<lastname>Vomacka</lastname>
     </author>
    </authors>
</bookstore>
```

Kolik je autorů?

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/
XSL/Transform" version="1.0">

  <xsl:template match="/">

   <xsl:value-of select="count(//author)"/>

  </xsl:template>

</xsl:stylesheet>
```

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

<bookstore>
  <!--Authors-->
  <authors>
     <author id="author01">
        <firstname>Josef</firstname>
        <lastname>Novak</lastname>
     </author>
     <author id="author02">
        <firstname>Jana</firstname>
        <lastname>Novotna</lastname>
     </author>
     <author id="author03">

<firstname>Vladislav</firstname>

<lastname>Vomacka</lastname>
     </author>
    </authors>
</bookstore>
```

Vypiš autorů a jejich id

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">

<xsl:template match="/">
    <xsl:apply-templates />
</xsl:template>

 <xsl:template match="//author">
  <div>
  jmeno: <xsl:value-of select="firstname" />
  prijmeni: <xsl:value-of select="lastname"/>
  id: <xsl:value-of select="@id" />
  </div>
 </xsl:template>
</xsl:stylesheet>
```

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

<bookstore>
  <!--Authors-->
  <authors>
     <author id="author01">
        <firstname>Josef</firstname>
        <lastname>Novak</lastname>
     </author>
     <author id="author02">
        <firstname>Jana</firstname>
        <lastname>Novotna</lastname>
     </author>
     <author id="author03">

<firstname>Vladislav</firstname>

<lastname>Vomacka</lastname>
     </author>
    </authors>
</bookstore>
```

Vypiš iniciály všech autorů

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
 <xsl:template match="/">
        <xsl:apply-templates />
 </xsl:template>


 <xsl:template match="//author">
  <div>
  jmeno: <xsl:value-of
select="substring(firstname,1,1)" />.
  prijmeni: <xsl:value-of
select="substring(lastname,1,1)"/>.
  </div>
 </xsl:template>
</xsl:stylesheet>
```

Příklad – seznam studentů

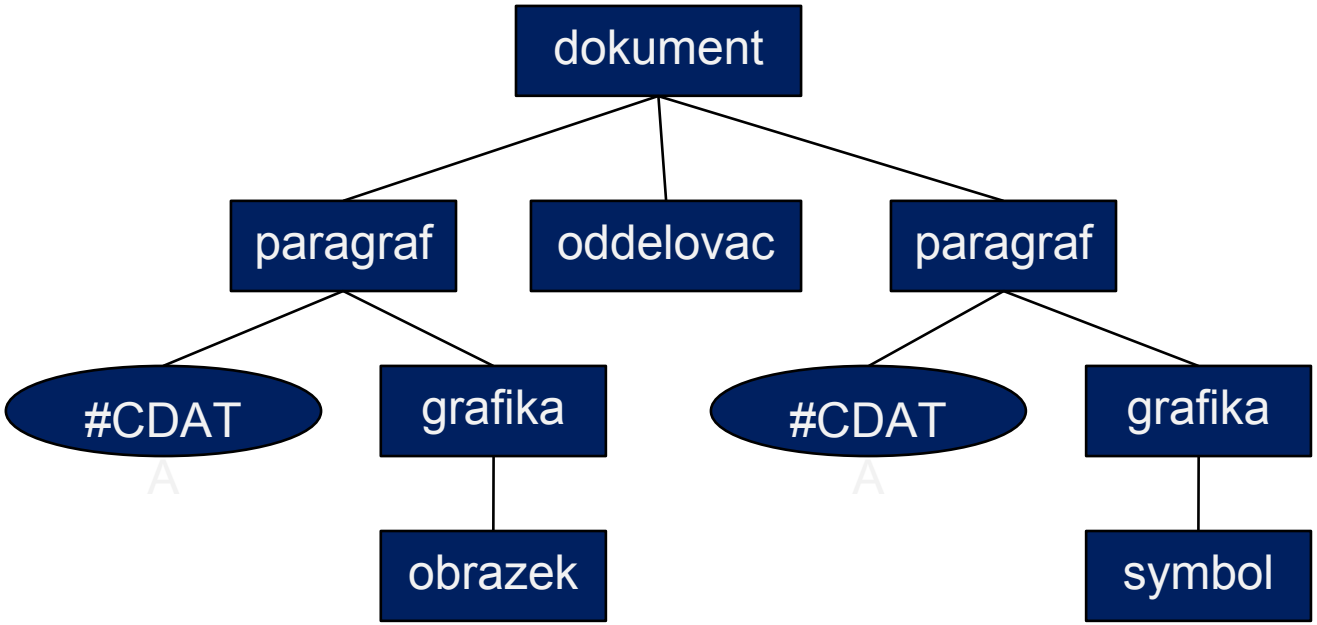
Zdrojový XML dokument

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

<studenti>
    <student>
    <jmeno>Martin</jmeno>
    <prijmeni>Klima</prijmeni>
    <adresa>
        <ulice>Horalkova 12</ulice>
        <mesto>Praha</mesto>
        <psc>120 00</psc>
    </adresa>
    </student>

    <student>
    <jmeno>Jaroslav</jmeno>
    <prijmeni>Vomacka</prijmeni>
    <adresa>
        <ulice>Neknubova 100</ulice>
        <mesto>Brno</mesto>
        <psc>602 00</psc>
    </adresa>
    </student>

</studenti>
```

Tento dokument chceme dostat do HTML tabulky

Napíšeme XSL transformaci, která to dokáže

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="studenti">
    <html>
    <body>
      <table border="1">
        <tr>
          <th>Jmeno</th>
          <th>Prijmeni</th>
          <th>Mesto</th>
        </tr>
        <xsl:apply-templates/>
      </table>
    </body>
    </html>
</xsl:template>

  <xsl:template match="student">
    <tr>
      <td>
      <xsl:value-of select="jmeno"/>
      </td>
      <td>
```


dokument
paragraf
oddelovac
paragraf
#CDAT
grafika
#CDAT
grafika
obrazek
symbol