



evropský
sociální
fond v ČR



EVROPSKÁ UNIE



MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY



OP Vzdělávání
pro konkurenceschopnost



TECHNICKÁ
UNIVERZITA
V LIBERCI
www.tul.cz

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Realizováno za finanční podpory ESF a státního rozpočtu ČR
v rámci v projektu *Zkvalitnění a rozšíření možností studia
na TUL pro studenty se SVP* reg. č. CZ.1.07/2.2.00/29.0011

Strom dokumentu

XML strom

- textový soubor je *zápisem* datového souboru
- zpracovávající software si zpravidla staví strom odpovídající XML datům – interní reprezentace odrážející vzájemné vztahy prvků
- výhodné pro posuzování struktury a manipulaci s ní

Příklad: XML dokument...

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

```
<diskoteka>
```

```
<disk id="d1">
```

```
  <interpret>Divokej Bill</interpret>
```

```
  <nazev>Propustka do pekel</nazev>
```

```
</disk>
```

```
<disk id="d2">
```

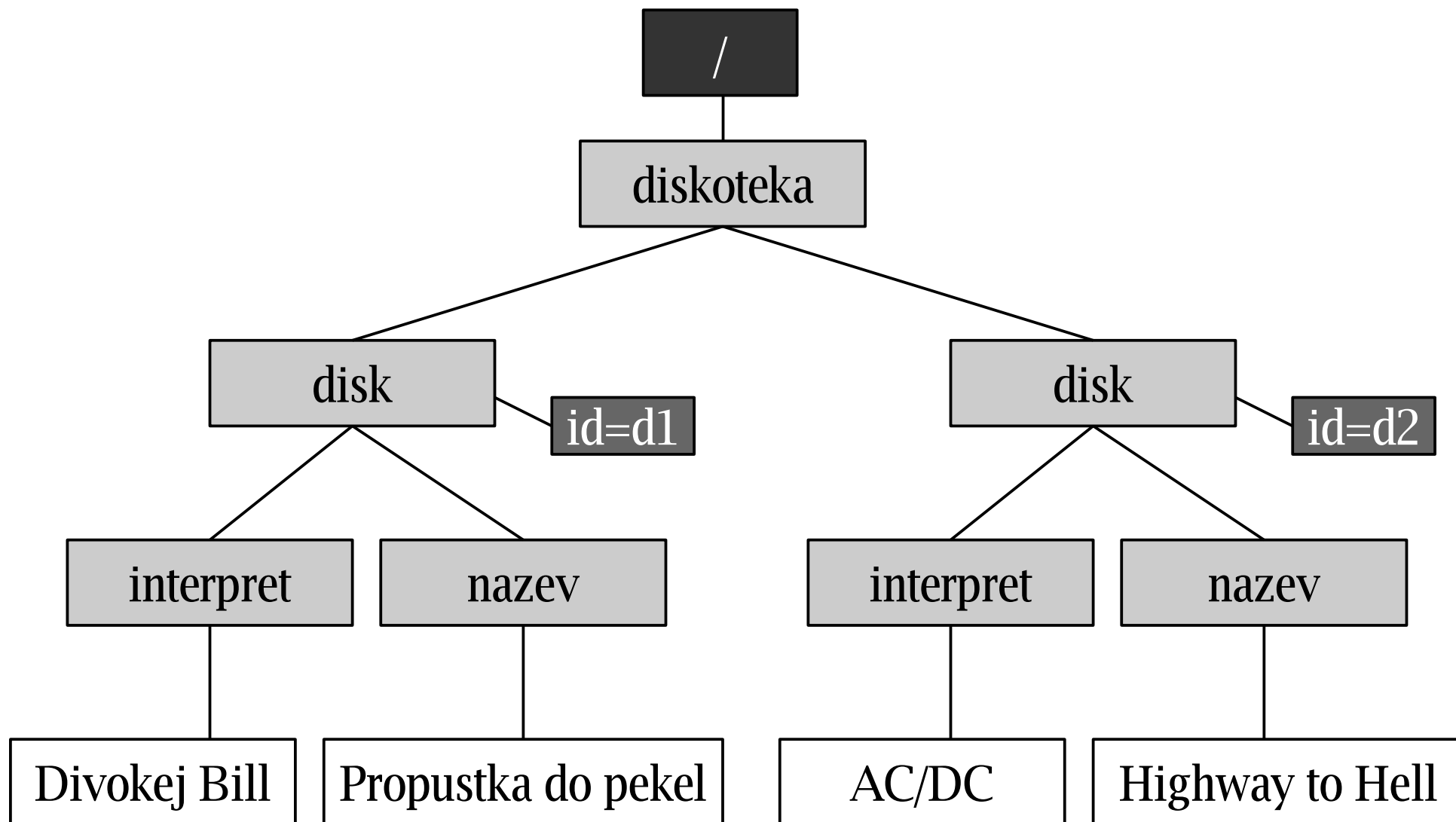
```
  <interpret>AC/DC</interpret>
```

```
  <nazev>Highway to Hell</nazev>
```

```
</disk>
```

```
</diskoteka>
```

...a jeho strom



Typy uzlů

- **kořen** – uměle přidaný
- **prvek** – odpovídají prvkům dokumentu
- **text** – textový obsah, bezejmenné, vždy listové
- **atribut** – připojen k prvku, který jej nese (ten je jeho rodičem, ale atribut není považován za dítě)
- **jmenný prostor** – název=prefix, dědí se
- **instrukce pro zpracování** – názvem je její cíl
- **komentář** – bezejmenný

Vztahy prvků

- **A je předkem B, B je potomkem A:**
 - B je obsažen v A
- **A je rodičem B, B je dítětem A:**
 - B je přímo obsažen v A (je ve struktuře právě o jednu úroveň níže)
 - rodič je vždy právě jeden
- **A je sourozencem B:**
 - mají stejného rodiče

XPath

XPath

- nástroj pro vyhledávání informací v XML dokumentech – identifikaci uzlů a jejich skupin ve stromě (prvků, atributů,...)
- řada (více než sto) vestavěných funkcí
- není samostatný jazyk
 - definuje syntax výrazů
 - používány jako součást dalších mechanismů (zejména XSLT, XQuery)

XPath

- nemá XML syntaxi
 - nepotřebuje – typicky tvoří hodnotu atributu
 - připomíná cestu v systému souborů
- **výsledkem** XPath výrazu je hodnota nebo skupina uzlů XML stromu vyhovujících podmínkám
- interpretace začíná ve **výchozím uzlu**
 - může/nemusí být kořen
 - typicky identifikuje uzly relativně vůči výchozímu

XPath cesta

- XPath cesta je sekvencí kroků, oddělovány lomítky
 - *krok1/krok2/.....*
- každý krok může mít tři části
 - **identifikátor osy** – směr procházení, výchozí množina
 - **test uzlu** – kterých uzlů se týká, povinná část
 - **podmínka (predikát)** – zužuje výsledky předchozího výběru
 - plný tvar: *osa::uzel[podmínka]*

Vyhodnocení cesty

- každý krok se vyhodnocuje v určitém **kontextu** – vychází se z konkrétního uzlu
 - první krok: z aktuálního nebo kořenového uzlu
 - další kroky: z uzlů vybraných předchozím krokem (postupně pro každý z nich)
- části kroku postupně omezují množinu vyhovujících uzlů: osa vybere základní množinu, test uzlu z ní vybere jen některé a případné podmínky ponechají jen ty, jež jim vyhovují

Oddělování kroků

- **znak /**

- prostý oddělovač
- díky implicitní ose **child::** se další krok typicky týká přímých potomků uzlů vybraných předchozím krokem
- na začátku cesty: absolutní cesta, začíná v kořeni

- **dvojice //**

- mezi uzly se může nacházet libovolný počet mezilehlých
- na začátku cesty: začíná se v kořeni, první prvek může být libovolně hluboko, trvá tak báječně dlouho...

Vybírání uzlů podle názvu

- nejjednodušší je uvést jméno prvku
 - **nazev** – nazev jako potomek aktuálního uzlu
- často požadujeme prvek v určitém kontextu, vyjádřen sekvencí kroků s příslušnými oddělovači
 - **/diskoteka/disk/nazev**
 - **/diskoteka//nazev** – nazev kdekoli uvnitř diskoteka
 - **//nazev** – nazev kdekoli v dokumentu
- ***** – vyhoví libovolný prvek

Vybírání uzlů podle typu

- lze požadovat libovolný uzel zadaného typu
- nezáleží na jménu, jen na charakteru
- dostupné typy:
 - `text()` – textový uzel, např. `nazev/text()`
 - `comment()` – uzel s komentářem
 - `processing-instruction()` – instrukce pro zpracování, v závorkách lze uvést i její požadované jméno
 - `node()` – libovolný uzel

Osy a podmínky

Identifikátor osy (1)

- **child::** děti aktuálního uzlu, implicitní osa
- **descendant::** všichni potomci aktuálního uzlu
- **descendant-or-self::** uzel sám a jeho potomci
- **parent::** rodič aktuálního uzlu
- **ancestor::** všichni předci aktuálního uzlu
- **ancestor-or-self::** uzel sám a jeho předci
- **self::** sám aktuální uzel

Identifikátor osy (2)

- **following-sibling::** následující sourozenci uzlu
- **preceding-sibling::** předchozí sourozenci uzlu
- **following::** uzly následující v dokumentu za aktuálním uzlem, kromě jeho potomků
- **preceding::** uzly předcházející v dokumentu před aktuálním uzlem, kromě jeho předků
- **attribute::** atributy aktuálního uzlu
- **namespace::** jmenné prostory aktuálního uzlu

Rozdělení dokumentu

- osy

self

ancestor

preceding

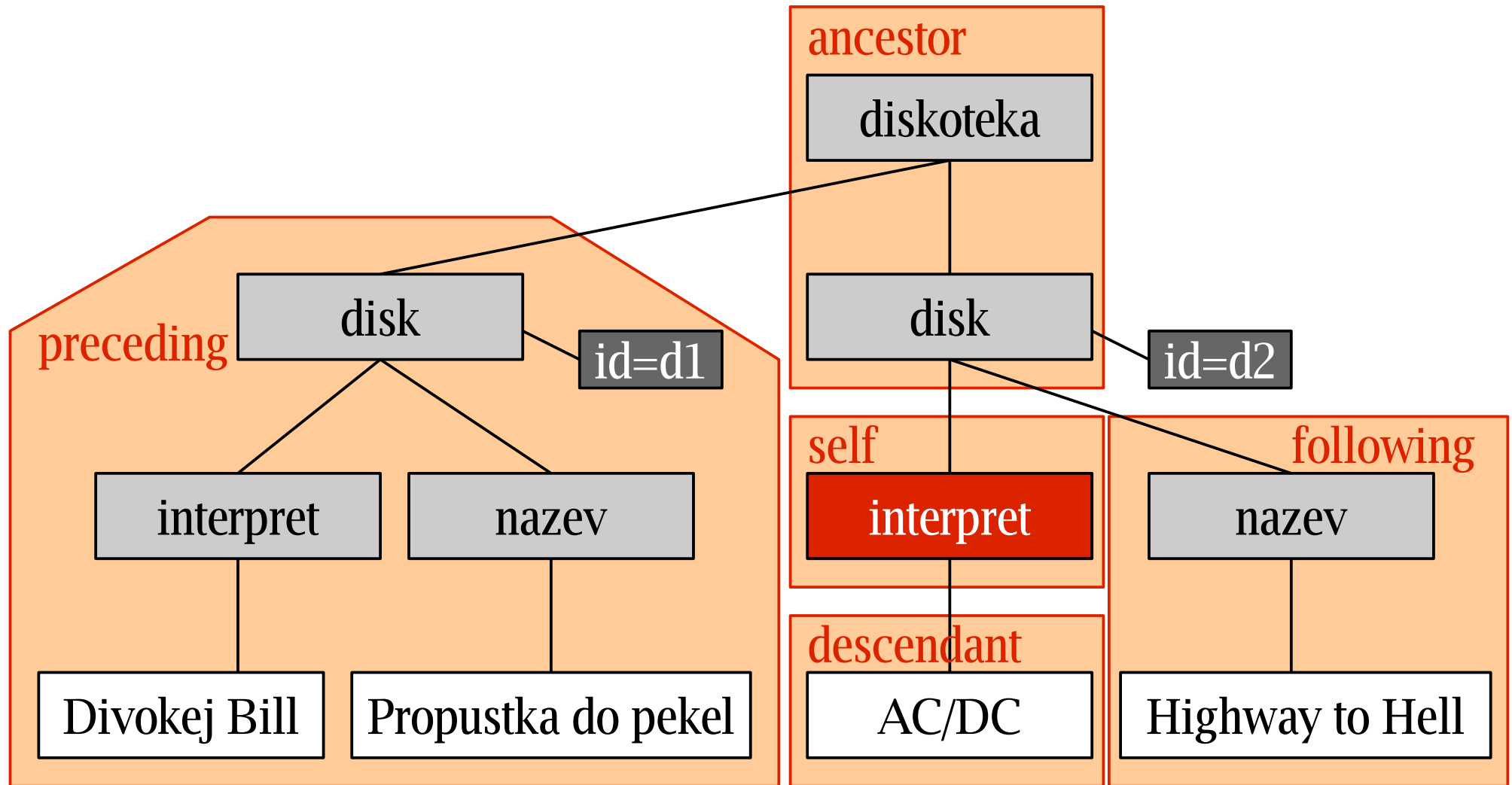
descendant

following

rozdělují dokument na pět navzájem disjunktních částí a kompletně jej pokrývají

- nezahrnují atributy a jmenné prostory

Příklad rozdělení



Zkratky

- **.** aktuální uzel **self::node()**
./navez názvy, které jsou potomky akt. uzlu
- **..** rodič **parent::node()**
../interpret interpret jako sourozenec
- **@** atribut **attribute::**
../@id rodičův atribut id
@* všechny atributy aktuálního prvku
- **//** lib. část cesty **/descendant-or-self::node()/**
//@id/.. všechny prvky obsahující atribut id

Podmínky

- v hranatých závorkách za testem uzlu
- **[*n*]** *n*-tý uzel v množině
 - ***[3]** ... třetí dítě aktuálního prvku
 - **(//nazev)[2]** ... druhý název v dokumentu
 - **following::*[1]** ... první prvek následující za aktuálním
 - osy mířící zpět (ancestor, preceding,...) číslují odzadu
 - **preceding::*[1]** ... nejbližší předchozí prvek
- častou podmínkou je hodnota atributu
 - **//disk[@medium="dvd"]**



Funkce



Funkce (1)

- **position()** ... číslo
 - pozice (pořadí, číslováno od 1) aktuálního uzlu v množině
 - [3] je zkratka za [position()=3]
- **last()** ... číslo
 - pozice posledního uzlu v množině
- **count(*množina_uzlů*)** ... číslo
 - počet uzlů v dané množině

Funkce (2)

- *id(objekt)* ... množina uzlů
 - vydá uzel nesoucí daný identifikátor
 - *id("d1")* ... prvek s identifikátorem „d1“
 - *id("d1")/nazev* ... jeho název
- *name(množina_uzlů?)* ... řetězec
- *local-name(množina_uzlů?)* ... řetězec
- *namespace-uri(množina_uzlů?)* ... řetězec
- vydají kvalifikované jméno, lokální jméno a URI jmenného prostoru daného (aktuálního) uzlu

Řetězcové funkce (1)

- **string(*objekt?*)** ... řetězec
 - převede svůj argument na řetězec znaků
 - prvky typicky převedeny na svůj textový obsah
 - neexistující hodnota: NaN; nekonečno: Infinity (–Infinity)
- **concat(*řetězec1, řetězec2,...*)** ... řetězec
 - spojí (zřetězí) své argumenty
- **string-length(*řetězec*)** ... číslo
 - počet znaků v řetězci

Hledání řetězců

- **contains(*kde*, *co*)** ... boolean
 - vydá true, pokud je řetězec *co* obsažen v řetězci *kde*
- **starts-with(*kde*, *co*)** ... boolean
 - vydá true, pokud řetězec *kde* začíná podřetězcem *co*
- **substring-before(*kde*, *co*)** ... řetězec
substring-after(*kde*, *co*) ... řetězec
 - vydá část řetězce *kde* před/za prvním výskytem řetězce *co* v něm; prázdný řetězec, pokud *kde* neobsahuje *co*

Řetězcové funkce (2)

- **substring(*řetězec*, *začátek*, *počet_znaků*?)** ... řetězec
 - vydá část řetězce začínající daným znakem a obsahující zadaný počet znaků (až do konce, pokud chybí)
- **normalize-space(*řetězec*?)** ... řetězec
 - vrátí řetězec s normalizovaným volným místem
- **translate(*kde*, *vzor*, *náhrada*)** ... řetězec
 - v řetězci kde změní znaky ze vzoru za nahrazující znaky
 - nahrazují se znaky, ne řetězce (viz tr z Unixu)

Logické funkce

- **boolean(*objekt*)** ... boolean
 - konverze (false=NaN, 0, prázdný řet. a prázdná množina)
- **true(), false()**
- **lang(*řetězec*)** .. boolean
 - má aktuální uzel atribut xml:lang s hodnotou odpovídající zadanému *řetězci*?
 - např. **lang("en")** pro **<p xml:lang="en-us">..**</p>** je true**
- **not(*boolean*)** ... boolean
- operátory (ne funkce) **and** a **or**

Číselné funkce

- **number(*objekt?*)** ... číslo
 - řetězec: přeskočí počáteční mezery, pokud následuje zápis čísla, výsledkem je toto číslo; jinak NaN
 - boolean: true=1, false=0
 - množina uzlů: převede se na řetězec voláním string()
- **sum(*množina_uzlů*)** ... číslo
 - uzly se převedou na čísla a následně sečtou

Aritmetické a relační operátory

- standardní sada aritmetických operací
 - $+$, $-$, $*$ v obvyklých významech
 - **div** je dělení *reálných* čísel (v plovoucí desetinné čárce)
 - **mod** je zbytek po celočíselném dělení
- obvyklé relační operátory
 - $=$, \neq
 - $>$, \geq , $<$, \leq

Zaokrouhlovací funkce

- **round(*číslo*)** ... číslo
 - nejbližší celé číslo
- **floor(*číslo*)** ... číslo
 - nejbližší vyšší celé číslo
- **ceiling(*číslo*)** ... číslo
 - nejbližší nižší celé číslo

Složitější podmínky

- podmínek může být několik
 - `//disk[@medium="dvd"][5]` ... páté DVD v dokumentu
 - vyhodnocují se zleva doprava
- podmínka se nemusí nutně vztahovat přímo k aktuálnímu uzlu
 - `//disk[.//nazev]` ... všechny prvky disk obsahující (libovolně hluboko zanořený) nazev
 - `//nazev[..//interpret]` ... všechny prvky nazev, které mají sourozence interpret

Kombinování XPath výrazů

- | v obvyklé roli „nebo“
 - výsledek je sjednocením množin uzlů vybraných jednotlivými výrazy
 - `//interpret | //nazev` ... všechny prvky interpret a nazev
 - pro hodnoty atributů je zpravidla vhodnější použít `or`:
`//cena[@jednotka="EUR" or @jednotka="USD"]` versus
`//cena[@jednotka="EUR"] | //cena[@jednotka="USD"]`

Další vývoj

XPath 2.0

- vyvíjeno společně s XQuery 1.0
- **podpora typů** – základní jednoduché typy podle XML Schema (19 typů) plus uzly
- každá hodnota má 2 složky:
 - vlastní hodnotu
 - typ
- **vše je sekvence**

Sekvence

- libovolně dlouhá posloupnost obsahující atomické hodnoty a uzly
 - záleží na pořadí
 - může obsahovat duplicity
- **uzel nebo atomická hodnota je ekvivalentní jednoprvkové sekvenci obsahující tuto položku**
- **sekvence jsou jednoúrovňové**
 - při vnoření se zploští – $(1, (2, 3))$ se převede na $(1, 2, 3)$

Složité výrazy

- **for cyklus:**

for \$cd in //disk return \$cd/nazev

- **podmíněný výraz:**

if (count(//disk)>100) then "moc" else "málo"

- **kvantifikátory:**

some \$cd in //disk satisfies \$cd/nazev="Cosi"

every \$cd in //disk satisfies \$cd/nazev="Cosi"

XPath 3.0

- standardizováno 2014
- plný funkcionální jazyk
 - funkce je hodnota jako každá jiná
 - může být argumentem a/nebo výsledkem jiné funkce
 - více o funkcionálních jazycích viz *Alternativní metody programování*