5

Vývoj Internetových Aplikací

AJAX, JSON, XML

Ing. Michal Radecký, Ph.D.

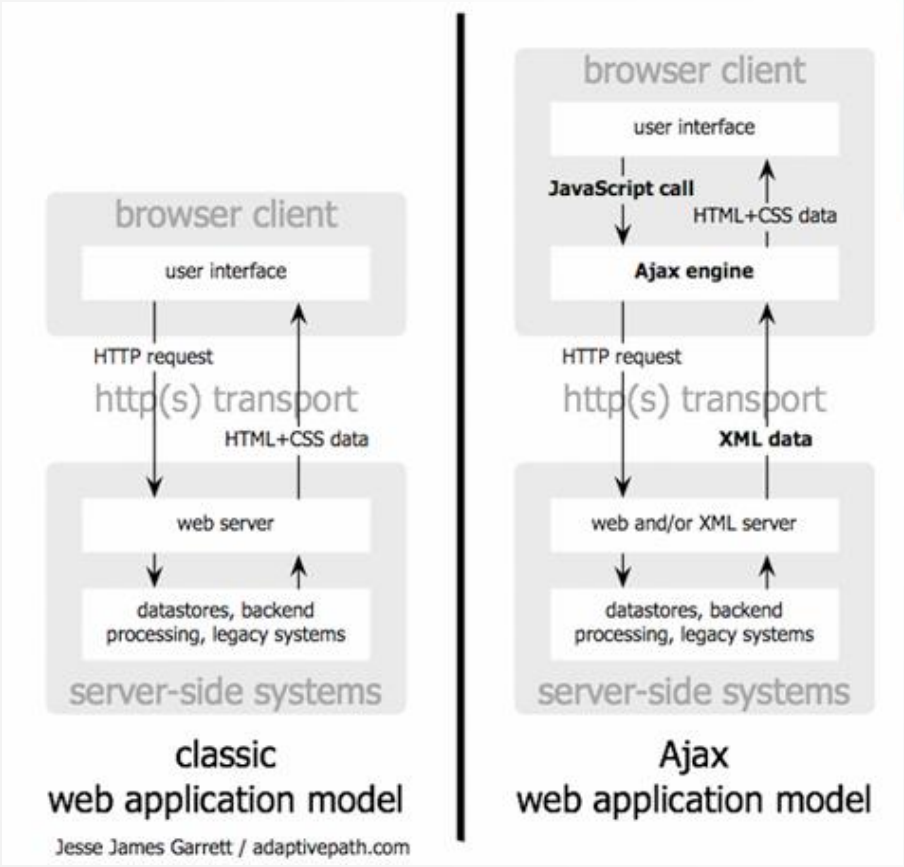
www.cs.vsb.cz/radecky

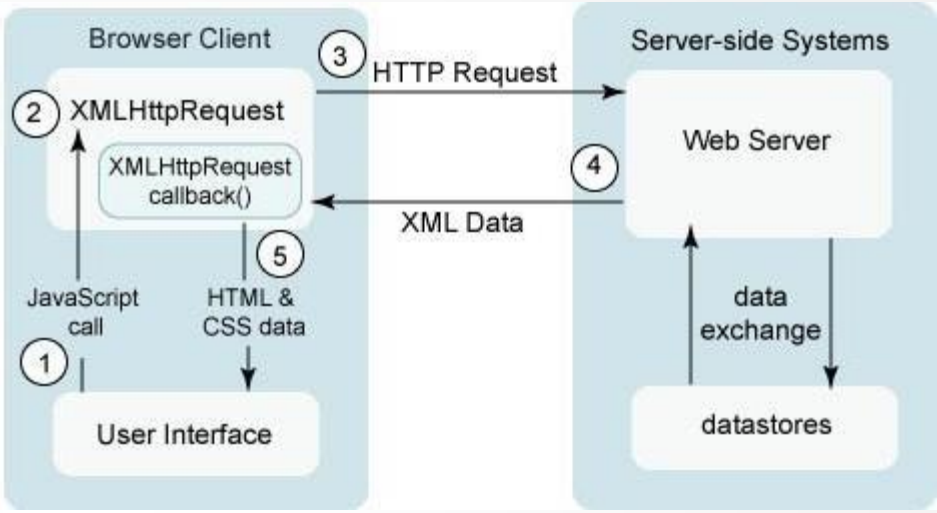
Co je to AJAX

- Asynchronous JavaScript and XML
- Kombinace technologií, která umožňuje měnit části webové stránky v závislosti na datech (vyvolání a zpracování HTTP požadavků), a to bez nutnosti aktualizace celé stránky.
- Vychází z dřívějších myšlenek (IFRAME, LAYER, Applety, apod.), v dnes používané podobě poprvé popsán v roce 2005
- Výhody
  - Větší uživatelský komfort a efektivita používání webových aplikací
  - Nižší nároky na množství přenesených dat
- Nevýhody
  - Eliminace funkčnosti tlačítka Zpět v prohlížeči
  - Změny uvnitř stránky neovlivňují stránku jako takovou (URL)

Model fungování

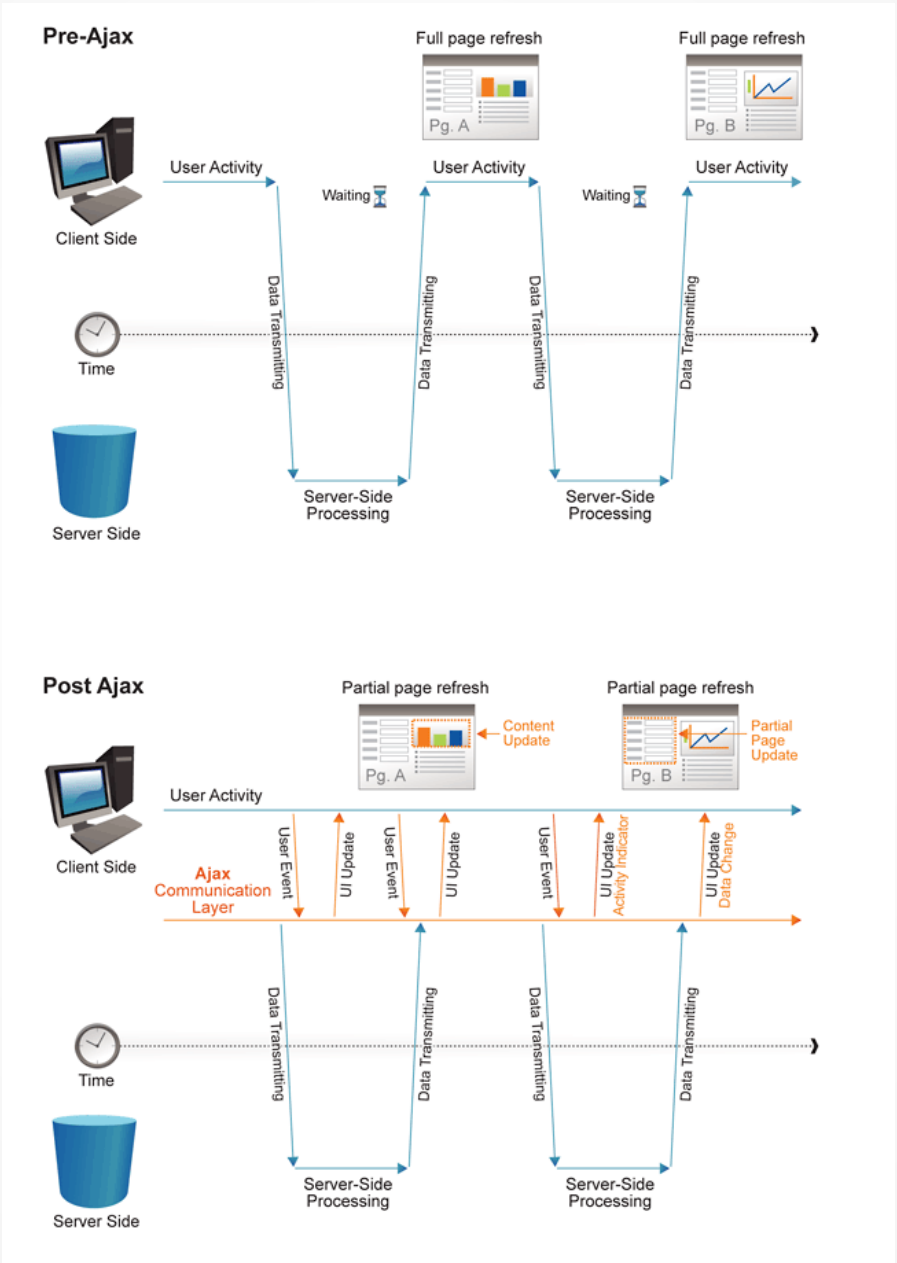
Zdroj: http://www.eioba.com





Model fungování

Zdroj: http://www.websiteoptimization.com



AJAX a implementace

Zdroj: http://blog.jur4.net/41-ajax-teoreticky-i-prakticky.html

- DOM a XmlHttpRequest
- Možnost využití frameworků (nejen Javascriptových, .NET, Java, Python, atd.)

```
if (window.XMLHttpRequest) {
   http_request = new XMLHttpRequest();
 } else if (window.ActiveXObject) {
    try {
      http_request = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (eror) {
      http_request = new ActiveXObject("Microsoft.XMLHTTP");
    }
 }
```

Vytvoření objektu

```
http_request.onreadystatechange = function() { zpracuj(http_request); };

http_request.open('POST', 'synonyma.php', true);
http_request.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
http_request.send(request);



function zpracuj(http_request) {
      if (http_request.readyState == 4) {
          if (http_request.status == 200) {
              alert(http_request.responseText);
          } else {
              alert('Chyba');
          }
      }
  }
```

Vyvolání AJAX dotazu

AJAX a jQuery

Zdroj: http://www.ibm.com

```
$('#stats').load('stats.html');
```

Nahrání HTML obsahu

```
$.post('save.cgi', {
    text: 'my string',
    number: 23
}, function() {
    alert('Your data has been saved.');
});
```

Zaslání dat na server (metodou POST)

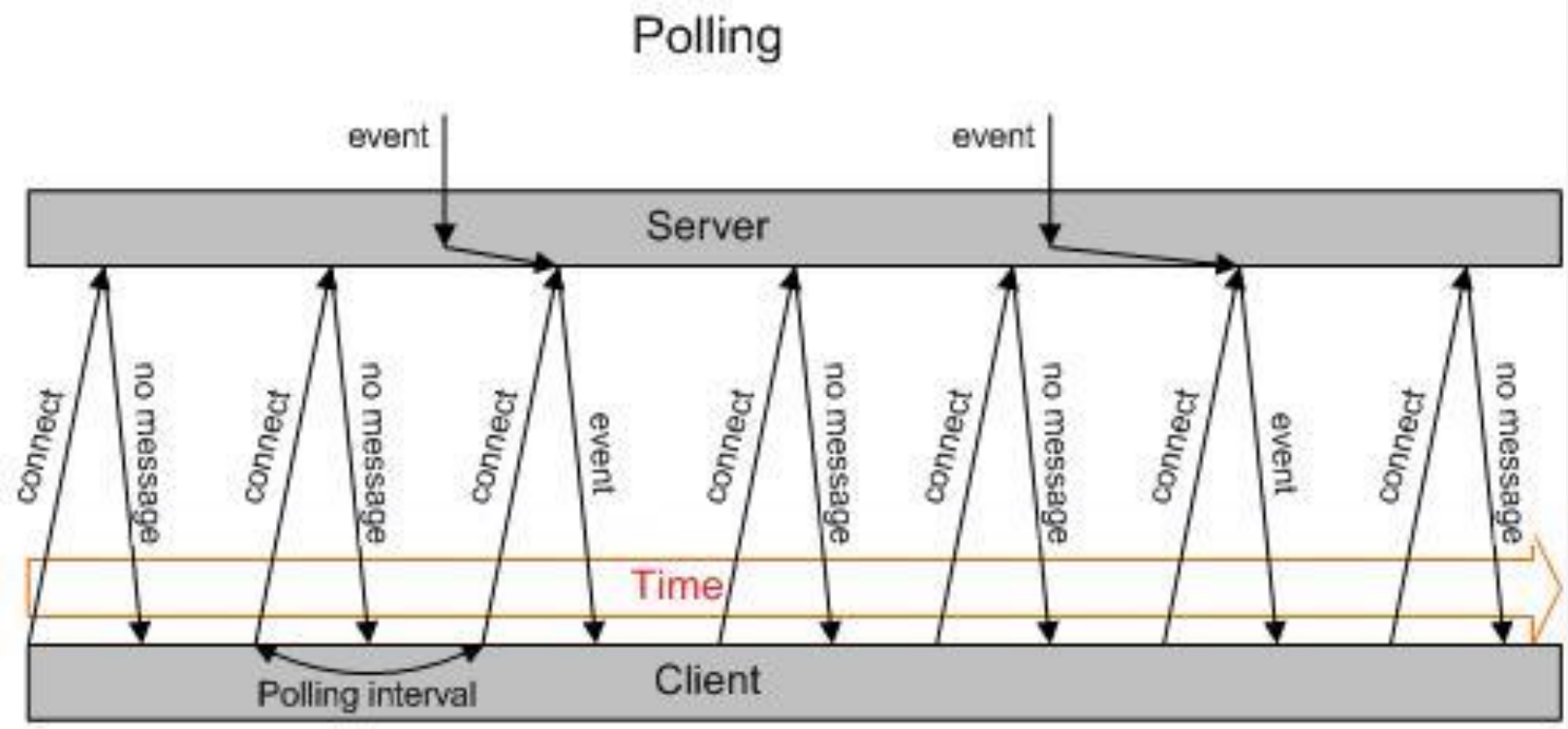
```
$.ajax({
    url: 'document.xml',
    type: 'GET',
    dataType: 'xml',
    timeout: 1000,
    error: function(){
        alert('Error loading XML document');
    },
    success: function(xml){
        $(xml).find('item').each(function(){
        var item_text = $(this).text();

        $('<li></li>')
            .html(item_text)
            .appendTo('ol');
    });
    }
});
```

Komplexní příklad zpracování XML jako AJAX dotazu

Asynchronní přístupy

Zdroj: http://pic.dhe.ibm.com/infocenter

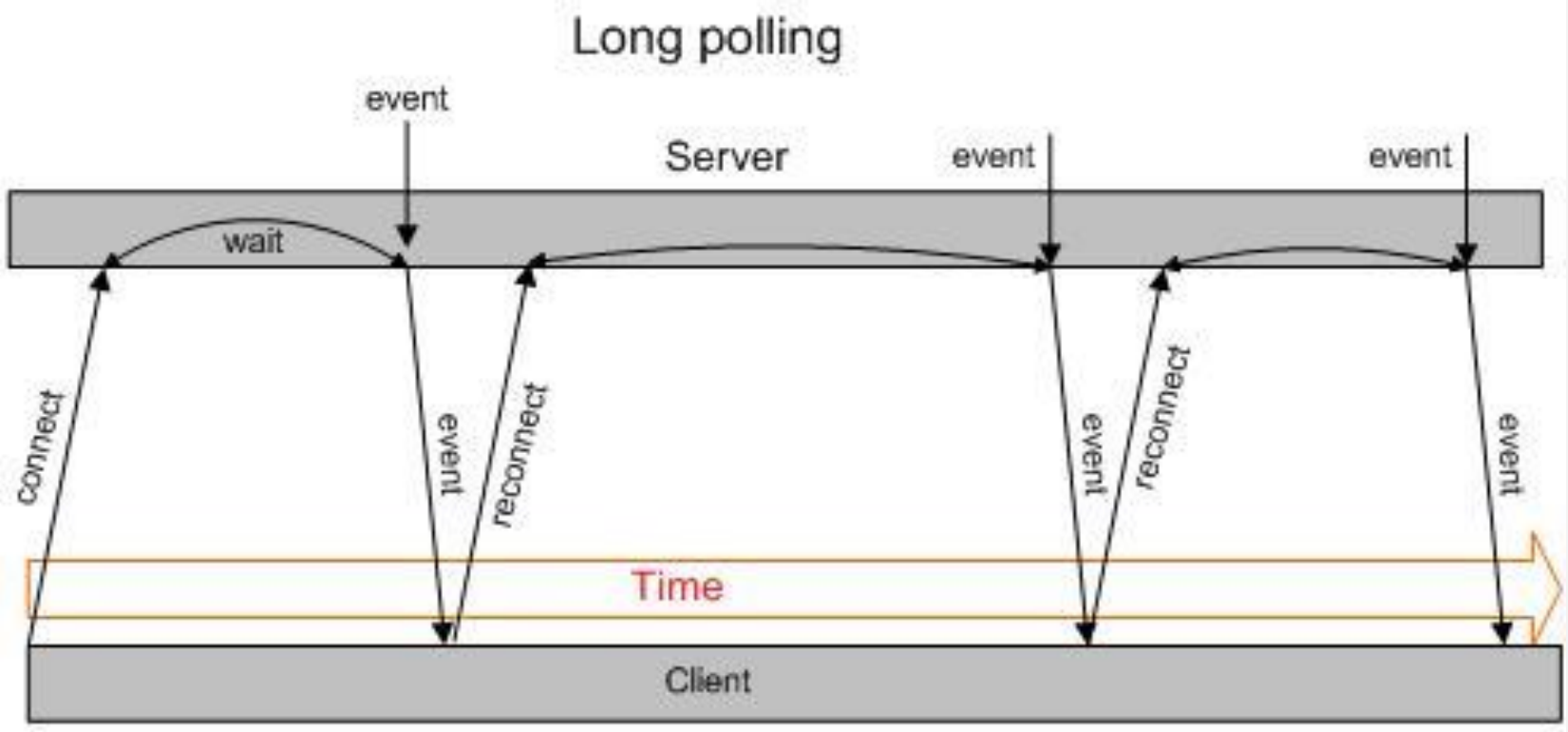
- Polling



Asynchronní přístupy

Zdroj: http://pic.dhe.ibm.com/infocenter

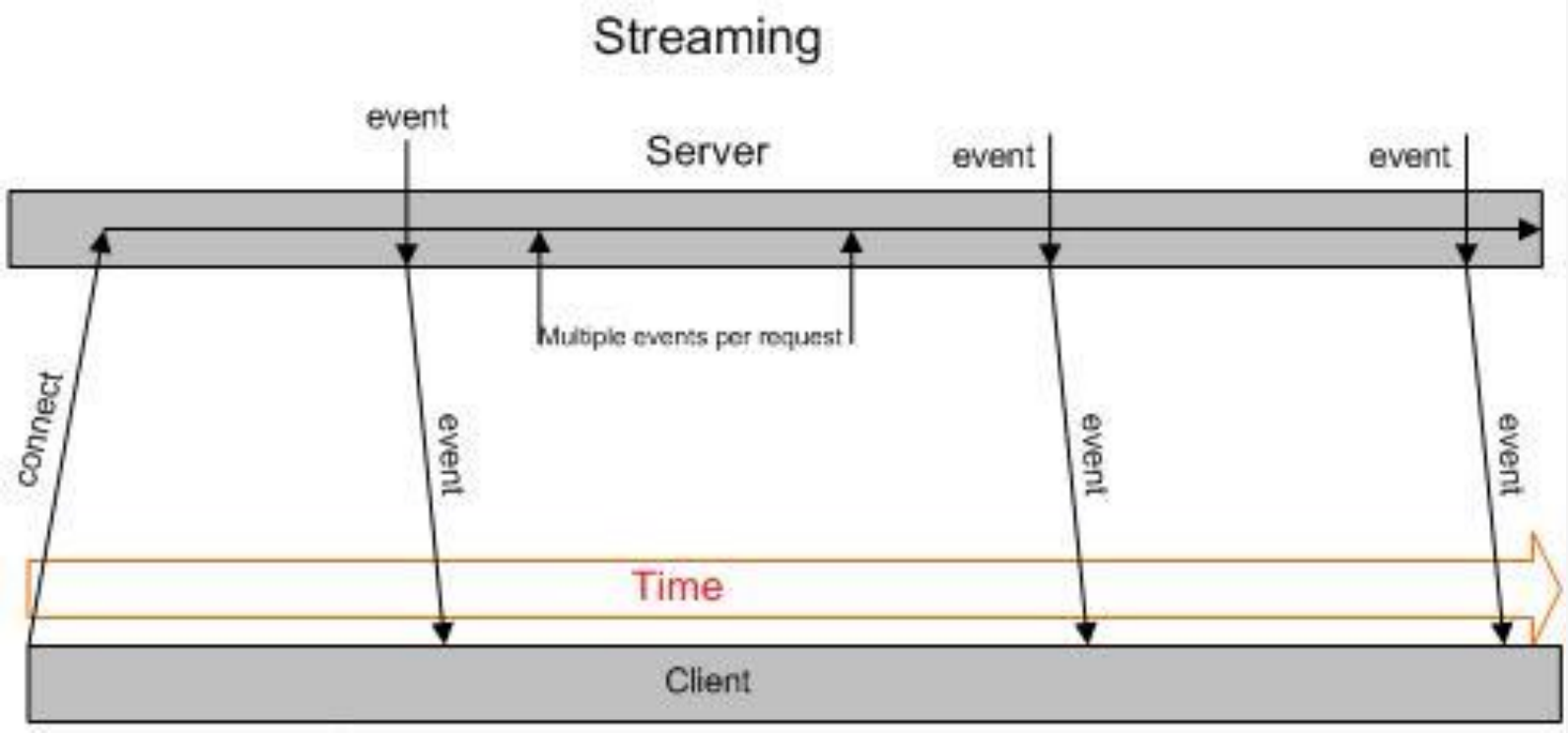
- Long - polling



Asynchronní přístupy

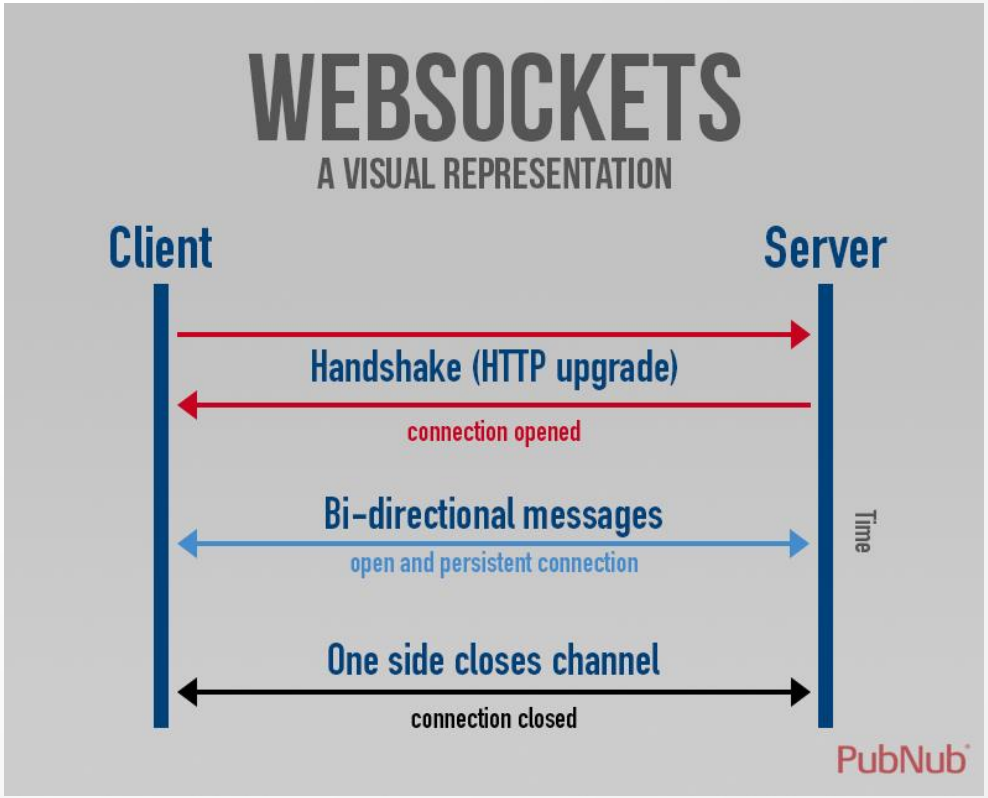
Zdroj: http://pic.dhe.ibm.com/infocenter

- Streaming
- Push přístup
- Comet, reverse AJAX – mnoho implementací, různé techniky



WebSockets

- Persistentní komunikační obosuměrný kanál
- Vlastní objekt WebSocket
- send, onmessage, onopen, onerror, readyState



Co je to XML

- eXtensible Markup Language
- množina pravidel
  - sémantické značky (tagy, elementy)
  - rozdělení dokumentu na části podle struktury
  - identifikace částí dokumentu
- jazyk pro popis jazyků
  - meta-značkový jazyk
  - definuje syntaxi definice jiného jazyka
- vychází se SGML (Standard Generalized Markup Language)
  - stejné možnosti
  - jednoduchost
- nejedná se o další značkovací jazyk
  - je to meta-jazyk
  - konkrétní názvy elementů, atributů, atd. jsou v režii tvůrce dokumentu

Proč používat XML

- data + značky = strukturovaná data se sémantikou
- umožňuje určení vazeb (vztahů) mezi elementy
- může být 100% ASCII text
- je detailně dokumentovaný W3C
- není patentovaný, nemá copyright a další podobná omezení
- neexistují verze XML (jako takového)
- podpora v programovacích jazycích
- podpora v nástrojích
- jednoduché zpracování

Formát XML

jsou rozšiřitelné
drží strukturu dokumentu

- Elementy/Tagy
  - značení odlišuje XML od čistého textu
  - většina značení jsou tagy (značky)
    - tag je vše co začíná znakem '<' a končí znakem '>'
    - tag má jméno
      - musí začínat `[a-z,A-Z,_]`
      - je case sensitive (`<B>` a `<b>` jsou různé)
  - prázdný tag
    - nemá obsah, může mít parametry
    - možnost použití zkratky pomocí koncovky '/>'

```
<empty />
<empty></empty>
```

  - pozor na znakové entity

```
<tag atribut="hodnota">
  data
</tag>
```

Znaková entita	znak
&amp;	&
&lt;	<
&gt;	>
&quot;	"
&apos;	'
&#37;	%
...	...

```
<section>
  <headline>Markup</headline>
  <text>
    Znaménka menší (&lt;)
    a ampersady (&amp;) jsou
    v normálním XML textu vždy
    zpracovány jako začátky
    tagu nebo entity.
  </text>
</section>
```

Formát XML

- Atributy
  - obsahují je počáteční a prázdné tagy
  - dvojice **`jméno = hodnota`**
  - jméno
    - musí začínat **`[a-z,A-Z,_]`**
    - stejné jméno v tagu jen jednou
  - hodnota
    - *řetězec* v uvozovkách (nebo apostrofech)
    - libovolné znaky
    - dodržování zanoření uvozovek resp. apostrofů

informace o dokumentu bez vztahu k obsahu dokumentu

možnost přidání informace bez změny stromové struktury dokumentu

Umístění dat

- data XML dokumentu mohou být
  - v atributech elementu
  - v obsahu elementu
- doporučení
  - vlastní data v elementech
  - informace o datech (meta-data) v atributech
  - do atributů obvykle
    - ID čísla
    - URL
    - informace ne přímo určené nebo důležité pro čtenáře

```
<activity creation="06/08/2000">
```

```
<activity>
  <creation day="08" month="06" year="2000" />
  ...
```

```
<activity>
  <creation>
    <day>08</day>
    <month>06</month>
    <year>2000</year>
  </creation>
  ...
```

Další specifikace

- Komentáře
  - začínají "**<!--**" a končí "**-->**"
- Text bez interpretace
  - sekce **CDATA**

```
<![CDATA[
for (int i = 0; i < array.length && error
== null; i++)
]]>
```

- Instrukce zpracování nadřazené aplikace
  - začínají "**<?nazev** " a končí "**?>**"

```
<?php echo "Hello world!"; ?>
```

- XML Prolog

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

- Specifikace MIME-type
  - application/xml, text/xml
  - application/mathml+xml, application/XSLT+xml, image/svg+xml

Jmenný prostor

- Jmenný prostor
  - slouží k oddělení různých množin specifikačních elementů pomocí prefixu
  - specifikace a použití pomocí **`xmlns:název`**
  - platnost pro podřazené elementy
  - definice NS odkazuje na URI (může být smyšlené nebo existující)

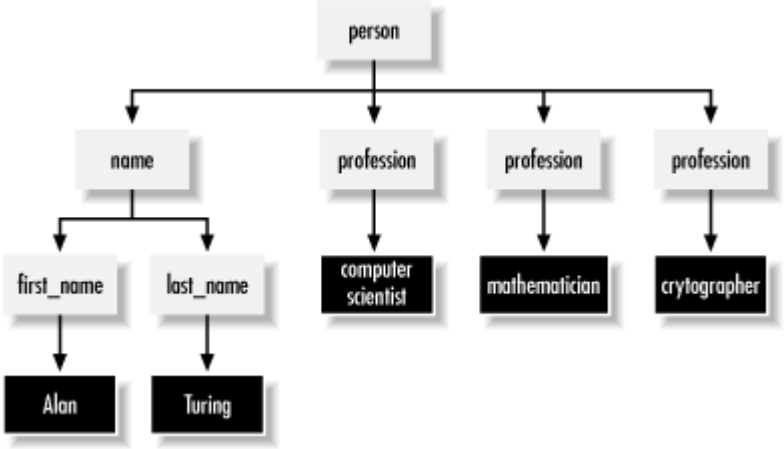
```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="keyword">
  ...
  </xsl:template>
</xsl:stylesheet>
```

```
<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform">
  <template match="keyword">
    <!-- undeclare default namespace -->
    <content-item xmlsn="">
      ...
```

Rodiče, děti, kořen, …

Zdroj: http://docstore.mik.ua/orelly/xml/xmlnut

- XML dokument odpovídá stromové struktuře
- Vždy musí být právě jeden kořen
- Elementy se nesmějí překrývat – křížit
- Vždy je možné definovat rodiče a děti každého elementu (rodič je vždy max. jeden, dětí může být 0 nebo více)




```
<person>
  <name>
    <first_name>Alan</first_name>
    <last_name>Turing</last_name>
  </name>
  <profession>computer scientist</profession>
  <profession>mathematician</profession>
  <profession>cryptographer</profession>
</person>
```

DTD

- Document Type Definition
- Specifikační jazyk pro popis pravidel a možností tvorby XML dokumentu
- Umožňuje korektní validaci XML dokumentu
- Určuje
  - seznam elementů, atributů, notací a entit
  - obsah elementů a atributů
  - vztahy mezi nimi
  - strukturu
- Nalézá se
  - v prologu za deklarací
  - před prvním elementem
- Buď přímo DTD nebo URL s DTD

```
<!DOCTYPE person[
  ...
]>
```

```
<!DOCTYPE person SYSTEM
  "http://abc.com/xml/dtds/person.dtd">
```

DTD – deklarace elementů

```
<!ELEMENT element_name content_specification>
```

- ANY
  - jakákoliv hodnota elementu je povolena (další elementy nebo #PCDATA)
- EMPTY
  - element nemá obsah
- (#PCDATA)
  - parsed character data, text pro parsování
- (child1, child2, …)
  - deklarace seznamu potomků – podřazené elementy
  - je možné používat regulární definice počtu (child1?, child2+, child3*)
- (child1 | child2)
  - deklarace možností potomků
- Pomocí závorek je možné deklarace kombinovat

```
<!ELEMENT name (last_name
               | (first_name, ( (middle_name+, last_name) | (last_name?) )
               ) >
```

DTD – deklarace atributů

```
<!ATTLIST element_name attribute_name
content_specification default_value>
```

- CDATA
  - zparsovatelný text
- NMTOKEN, NMTOKENS
  - hodnota odpovídající specifikaci jména (bez mezer, atd.), např. name v HTML
- (pondělí | úterý | středa)
  - výčet přesně specifikovaných povolených hodnot
- ID
  - jedinečná identifikace v celém dokumentu, hodnota podle specifikace jména, element může mít max. jeden atribut tohoto typu
- IDREF, IDREFS
  - atribut pro specifikaci vazby na element s ID atributem
- ENTITY, ENTITIES
  - hodnota odkazuje na specifikovanou entitu

- „value“
  - konkrétní hodnota
- #IMPLIED
  - atribut je volitelný
- #REQUIRED
  - atribut je povinný
- #FIXED “value”
  - pokud atribut je, musí mít uvedenou hodnotu

DTD – deklarace entit

```
<!ENTITY entity_name content_specification>
```

- „value“
  - jedna konkrétní hodnota
- SYSTEM „url externího zdroje“
  - hodnota Entity je specifikována externě

```
<!DOCTYPE report [
  <!NOTATION eps SYSTEM "text/postscript">
  <!ENTITY logo SYSTEM "logo.eps" NDATA eps>
  <!ELEMENT image EMPTY>
  <!ATTLIST image source ENTITY #REQUIRED>
  ...
]>
<report>
  <!-- general entity reference (invalid) -->
  &logo;
  ...
  <!-- attribute value -->
  <image source="logo" />
</report>
```

DTD a XML příklad

Zdroj: http://www.idevelopment.info/data/Programming/java/xml/ExampleXMLandDTDFile.html

XML

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

<DatabaseInventory>

 <DatabaseName>
  <GlobalDatabaseName>production.iDevelopment.info</GlobalDatabaseName>
  <OracleSID>production</OracleSID>
  <DatabaseDomain>iDevelopment.info</DatabaseDomain>
  <Administrator EmailAlias="jhunter" Extension="6007">Jeffrey Hunter</Administrator>
  <DatabaseAttributes Type="Production" Version="9i"/>
  <Comments>
   The following database should be considered the most stable
   up-to-date data. The backup strategy includes running the dat
   in Archive Log Mode and performing nightly backups. All new
   need to be approved by the DBA Group before being created.
  </Comments>
 </DatabaseName>

 <DatabaseName>
  <GlobalDatabaseName>development.iDevelopment.info</Glob
  <OracleSID>development</OracleSID>
  <DatabaseDomain>iDevelopment.info</DatabaseDomain>
  <Administrator EmailAlias="jhunter" Extension="6007">Jeffrey
  <Administrator EmailAlias="mhunter" Extension="6008">Melo
  <DatabaseAttributes Type="Development" Version="9i"/>
  <Comments>
   The following database should contain all hosted applications.
   data will be exported on a weekly basis to ensure all developn
   have stable and current data.
  </Comments>
 </DatabaseName>

 <DatabaseName>
  <GlobalDatabaseName>testing.iDevelopment.info</GlobalData
  <OracleSID>testing</OracleSID>
  <DatabaseDomain>iDevelopment.info</DatabaseDomain>
  <Administrator EmailAlias="jhunter" Extension="6007">Jeffrey
  <Administrator EmailAlias="mhunter" Extension="6008">Melo
  <Administrator EmailAlias="ahunter">Alex Hunter</Administra
  <DatabaseAttributes Type="Testing" Version="9i"/>
  <Comments>
   The following database will host more than half of the testing
   for our hosting environment.
  </Comments>
 </DatabaseName>

</DatabaseInventory>
```

DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT DatabaseInventory (DatabaseName+)>
<!ELEMENT DatabaseName (   GlobalDatabaseName
                         , OracleSID
                         , DatabaseDomain
                         , Administrator+
                         , DatabaseAttributes
                         , Comments)
>
<!ELEMENT GlobalDatabaseName (#PCDATA)>
<!ELEMENT OracleSID          (#PCDATA)>
<!ELEMENT DatabaseDomain     (#PCDATA)>
<!ELEMENT Administrator      (#PCDATA)>
<!ELEMENT DatabaseAttributes EMPTY>
<!ELEMENT Comments           (#PCDATA)>

<!ATTLIST Administrator       EmailAlias CDATA #REQUIRED>
<!ATTLIST Administrator       Extension  CDATA #IMPLIED>
<!ATTLIST DatabaseAttributes  Type       (Production|Development|Testing)
#REQUIRED>
<!ATTLIST DatabaseAttributes  Version    (7|8|8i|9i) "9i">

<!ENTITY AUTHOR "Jeffrey Hunter">
<!ENTITY WEB    "www.iDevelopment.info">
<!ENTITY EMAIL  "jhunter@iDevelopment.info">
```

XML Schema Definition (XSD)

- Nevýhody DTD
  - neřeší a nepodporuje jmenné prostory
  - neumožňuje specifikaci datových typů pro obsahy elementů a atributů
  - samotné DTD není XML
- XML Schema
  - specifikační jazyk založený na XML
  - pod hlavičkou W3C
  - definuje
    - strukturu XML dokumentu
    - elementy a atributy v XML dokumentu
    - definuje dětské elementy, jejich počet a pořadí
    - obsah elementu (prázdný/neprázdný)
    - datové typy elementů a atributů (přes 40 typů)
    - defaultní a pevné hodnoty elementů a atributů
  - vše ve specifikaci je v NS xs:

XSD

Zdroj: http://www.kosek.cz

```
<?xml version="1.0" encoding="utf-8"?>
<zamestnanci>
  <zamestnanec id="101">
    <jmeno>Jan</jmeno>
    <prijmeni>Novák</prijmeni>
    <email>jan@novak.cz</email>
    <email>jan.novak@firma.cz</email>
    <plat>25000</plat>
    <narozen>1965-12-24</narozen>
  </zamestnanec>
  <zamestnanec id="102">
    <jmeno>Petra</jmeno>
    <prijmeni>Procházková</prijmeni>
    <email>prochazkovap@firma.cz</email>
    <plat>27500</plat>
    <narozen>1974-13-21</narozen>
  </zamestnanec>
</zamestnanci>
```

XML

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="zamestnanci">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="zamestnanec"
                    maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="jmeno" type="xs:string"/>
              <xs:element name="prijmeni" type="xs:string"/>
              <xs:element name="email" type="xs:string"
                          maxOccurs="unbounded"/>
              <xs:element name="plat" type="xs:decimal"
                          minOccurs="0"/>
              <xs:element name="narozen" type="xs:date"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:int"
                          use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

W3C XML Schema

```
<!ELEMENT zamestnanci (zamestnanec+)>
<!ELEMENT zamestnanec (jmeno, prijmeni, email+,
                       plat?, narozen)>
<!ELEMENT jmeno       (#PCDATA)>
<!ELEMENT prijmeni    (#PCDATA)>
<!ELEMENT email       (#PCDATA)>
<!ELEMENT plat        (#PCDATA)>
<!ELEMENT narozen     (#PCDATA)>
<!ATTLIST zamestnanec
          id     CDATA #REQUIRED>
```

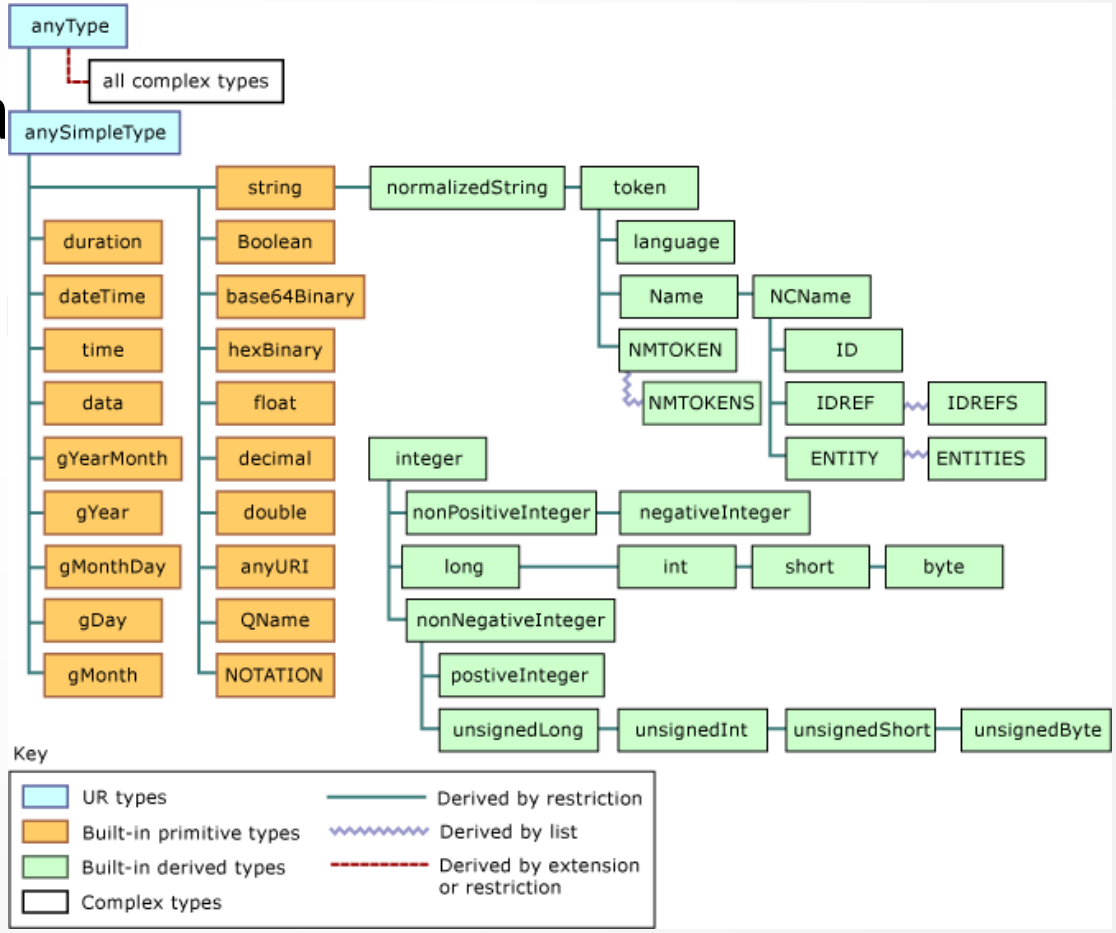
DTD

XSD – deklarace elementů

```
<xs:element name=„name" type=„type"/>
```

simple element

- Jméno odpovídá zvyklostem při vytváření jmen a proměnných
- Typ jeden z mnoha dostupných typů
- Možnost odvozová vlastních dat. typů




```
<xs:simpleType name="jménoType">
  <xs:restriction base="xs:string">
    <xs:minLength value="1"/>
    <xs:maxLength value="15"/>
  </xs:restriction>
</xs:simpleType>
```

```
<xs:simpleType name=„currencyType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="CZK"/>
    <xs:enumeration value="EUR"/>
    <xs:enumeration value="USD"/>
  </xs:restriction>
</xs:simpleType>
```

XSD – deklarace atributů

- Samotný atribut je simple-element jako součást complex-elementu

complex element

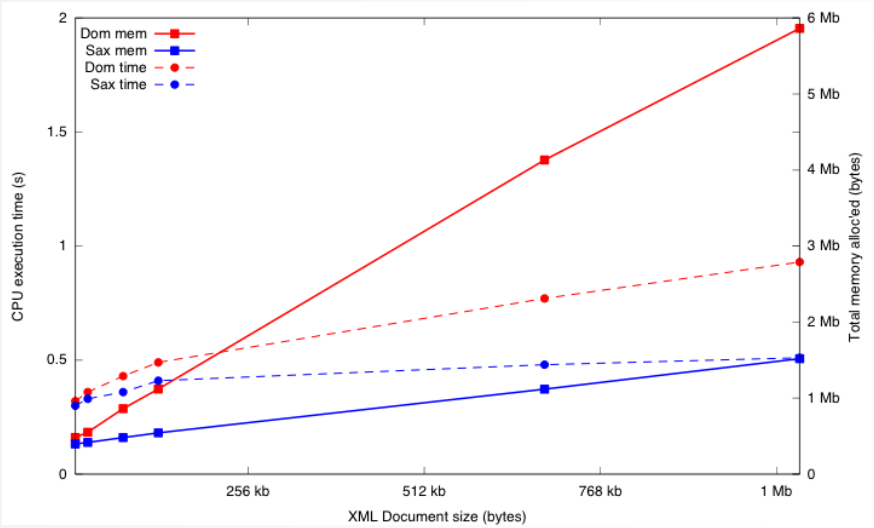
```
<xs:element name=„name“>
   <xs:complexType>
      <xs:sequence>
               <xs:element …/>
      </xs:sequence>
      <xs:attribute name=„name" type=„type"
                              use="required"/>
   </xs:complexType>
</xs:element>
```

Programová API

- DOM
  - Document Object Model
  - reprezentuje stromovou strukturu XML dokumentu v paměti pomocí objektů
  - je standardní rozhraní pro aplikace a práci s XML podle W3C
  - vyšší nároky na čas a paměť
- SAX
  - Simple API for XML – event-driven model
  - zpracování XML přímo při jeho načítání
  - zpracování formou volání metod/zpracování načtených dat na začátku a konci elementů, při zpracování textu, apod.
  - není přímo standardem
  - rychlý, vyšší nároky na samotnou aplikaci
- Parser
  - aplikace, program, třída, algoritmus
  - jeho úkolem je zpracovat XML dokument v textové podobě a převést jej do podoby dále zpracovatelné (např. DOM)
  - ověřuje syntaxi, zajišťuje ověření DTD a validaci

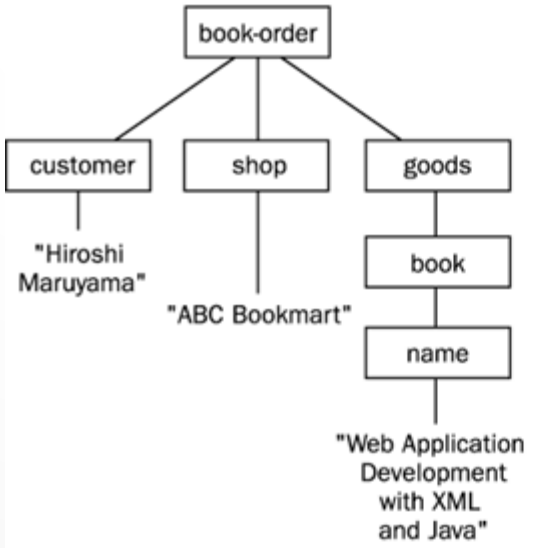
DOM vs. SAX

Zdroj: http://tech.inhelsinki.nl/2007-08-29/, http://book.javanb.com




```
<?xml version="1.0"encoding="utf-8"?>
<book-order>
  <customer>Hiroshi Maruyama</customer>
  <shop>ABC Bookmart</shop>
  <goods>
    <book>
      <name>Web Application Development with
XML and Java</name>
    </book>
  </goods>
</book-order>
```

Parsing




```
startElement: book-order
startElement: customer
characters: Hiroshi Maruyama
endElement: customer
startElement: shop
characters: ABC Bookmart
endElement: shop
startElement: goods
startElement: book
startElement: name
characters: Web Application
    Development with XML and Java
endElement: name
endElement: book
endElement: goods
endElement: book-order
```

DOM SAX

JavaScript

- Převod XML do DOMu

```
const xmlStr = '<q id="a"><span id="b">hey!</span></q>';
const parser = new DOMParser();
const doc = parser.parseFromString(xmlStr, "application/xml");

const errorNode = doc.querySelector("parsererror");

if (errorNode) {
  console.log("error while parsing");
} else {
  console.log(doc.documentElement.nodeName);
}
```

```
const xhr = new XMLHttpRequest();

xhr.onload = () => {
  dump(xhr.responseXML.documentElement.nodeName);
};

xhr.onerror = () => {
  dump("Error while getting XML.");
};

xhr.open("GET", "example.xml");
xhr.responseType = "document";
xhr.send();
```

JavaScript

- Práce v XMLDocument
  - Jako v rámci standardního DOMu webové stránky
  - Vstupem je XMLDocument (Document)
  - querySelector, querySelectorAll, getElement…
  - createTextNode, createElement, appendChild, …

```
const serializer = new XMLSerializer();
const xmlStr = serializer.serializeToString(doc);
```

  - Lze i validovat vůči DTD nebo XSD
    https://vegibit.com/how-to-parse-and-generate-xml-in-javascript/

XPath - Cesta

- Cesta (Path Expression) je hlavní konstrukční prvek pro specifikaci dotazů
- Obdoba definici cesty v souborovém systému OS
- Sekvence kroků oddělených „/“ nebo „//“
- Spojování více sekvencí s vazbou NEBO pomocí „|“
- Každý krok sestává z
  - identifikátoru osy (axes)
  - testu uzlu (povinný prvek)
  - predikátu
- Cesta se vyhodnocuje zleva doprava, a to relativně k aktuálnímu uzlu

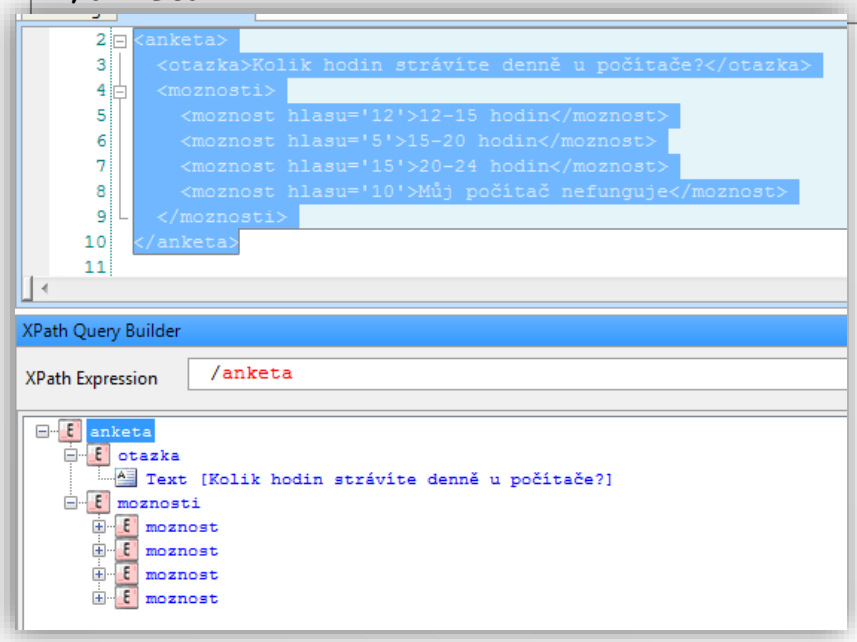
```
axisname::nodetest[predicate]
```

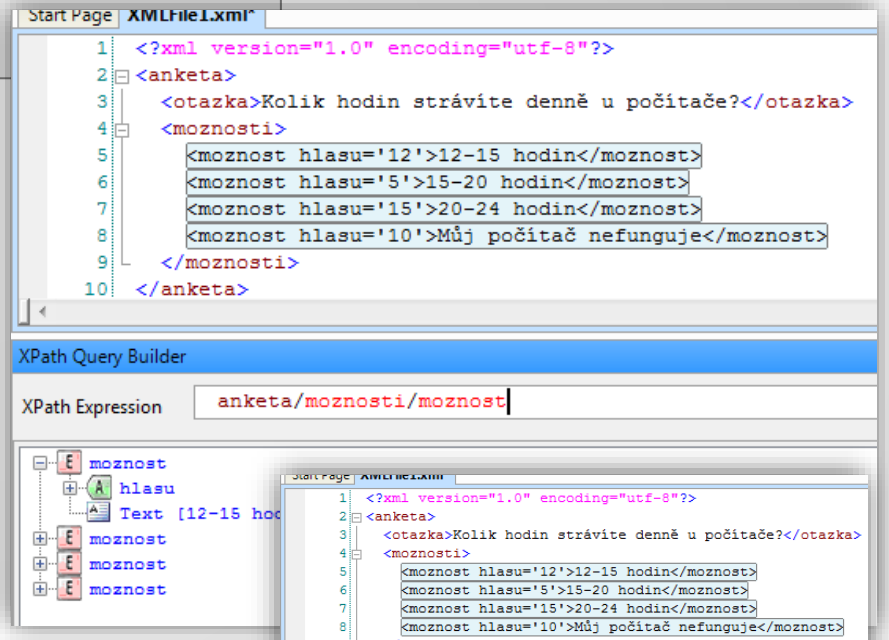
XPath – oddělovaní kroků

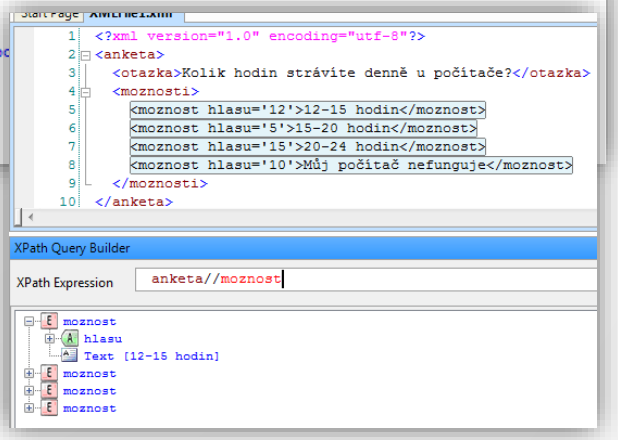
Zdroj: http://interval.cz/clanky/zaklady-jazyka-xpath/

Zdrojový XML soubor

```
<anketa>
  <otazka>Kolik hodin strávíte denně u počítače?</otazka>
  <moznosti>
    <moznost hlasu='12'>12-15 hodin</moznost>
    <moznost hlasu='5'>15-20 hodin</moznost>
    <moznost hlasu='15'>20-24 hodin</moznost>
    <moznost hlasu='10'>Můj počítač nefunguje</moznost>
  </moznosti>
</anketa>
```



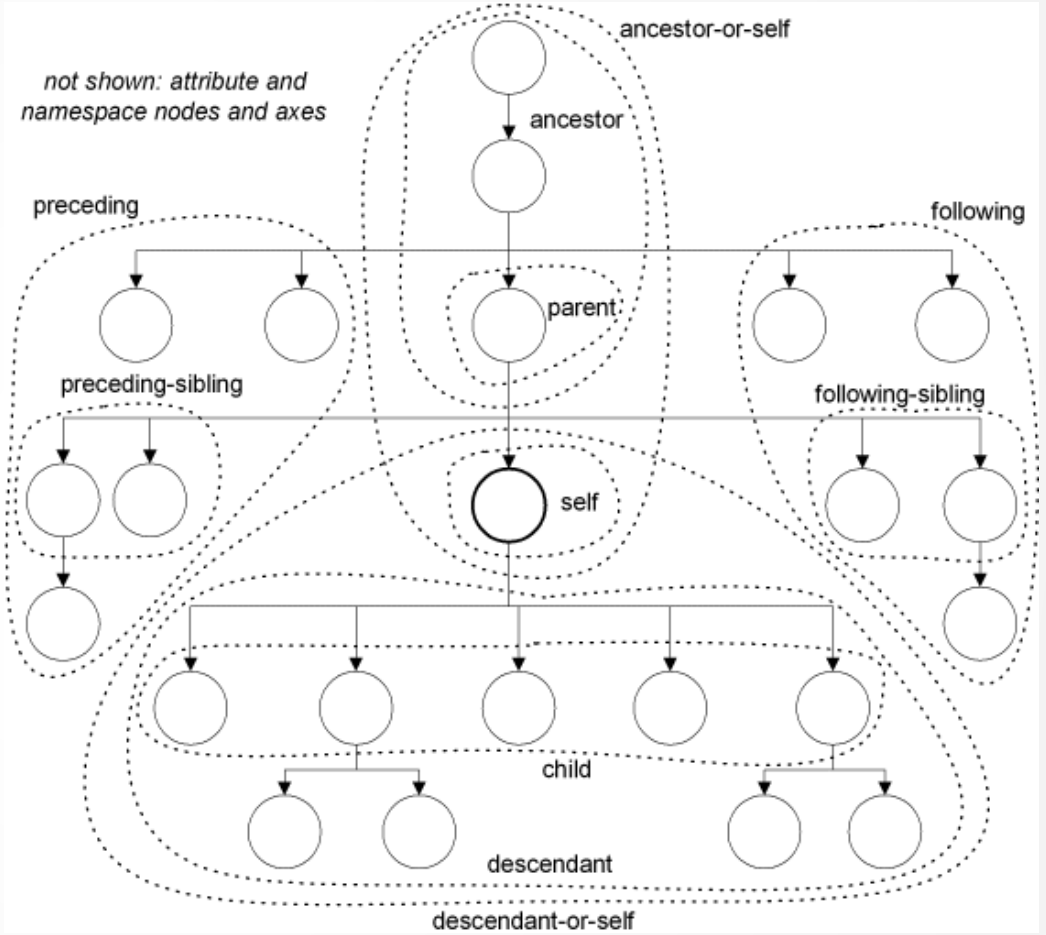




XPath - Osy

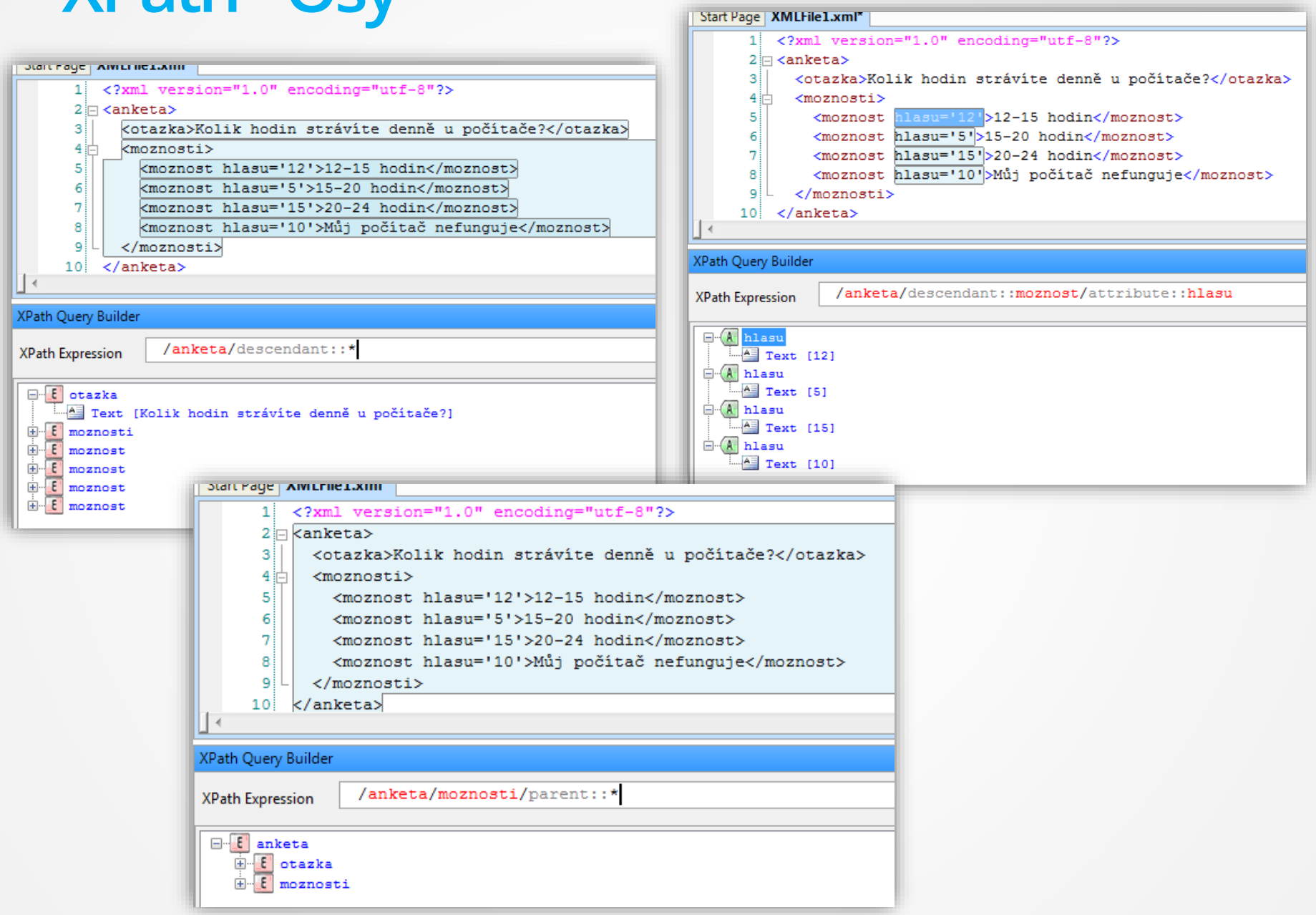
Zdroj: http://www.georgehernandez.com

- Specifikuje směr prohledávání XML stromu
- Množinu relevantních uzlů vstupujících do testování určuje právě specifikace osy, pokud se neuvede, pak child::
- Osy *ancestor*, *descendant*, *following*, *preceding* a *self* se nepřekrývají a dohromady obsahují všechny uzly dokumentu



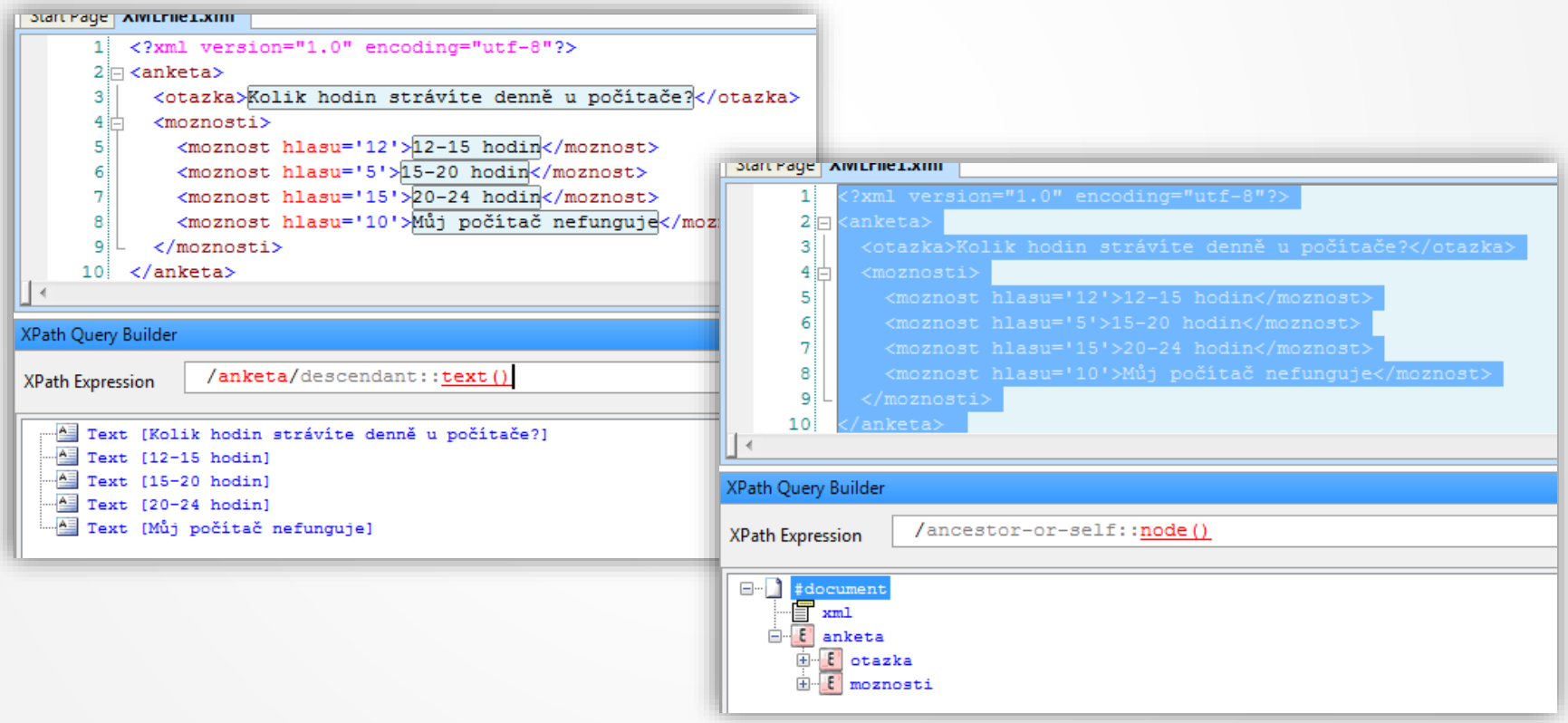
XPath - Osy

Zdroj: http://www.georgehernandez.com



XPath – testy uzlu

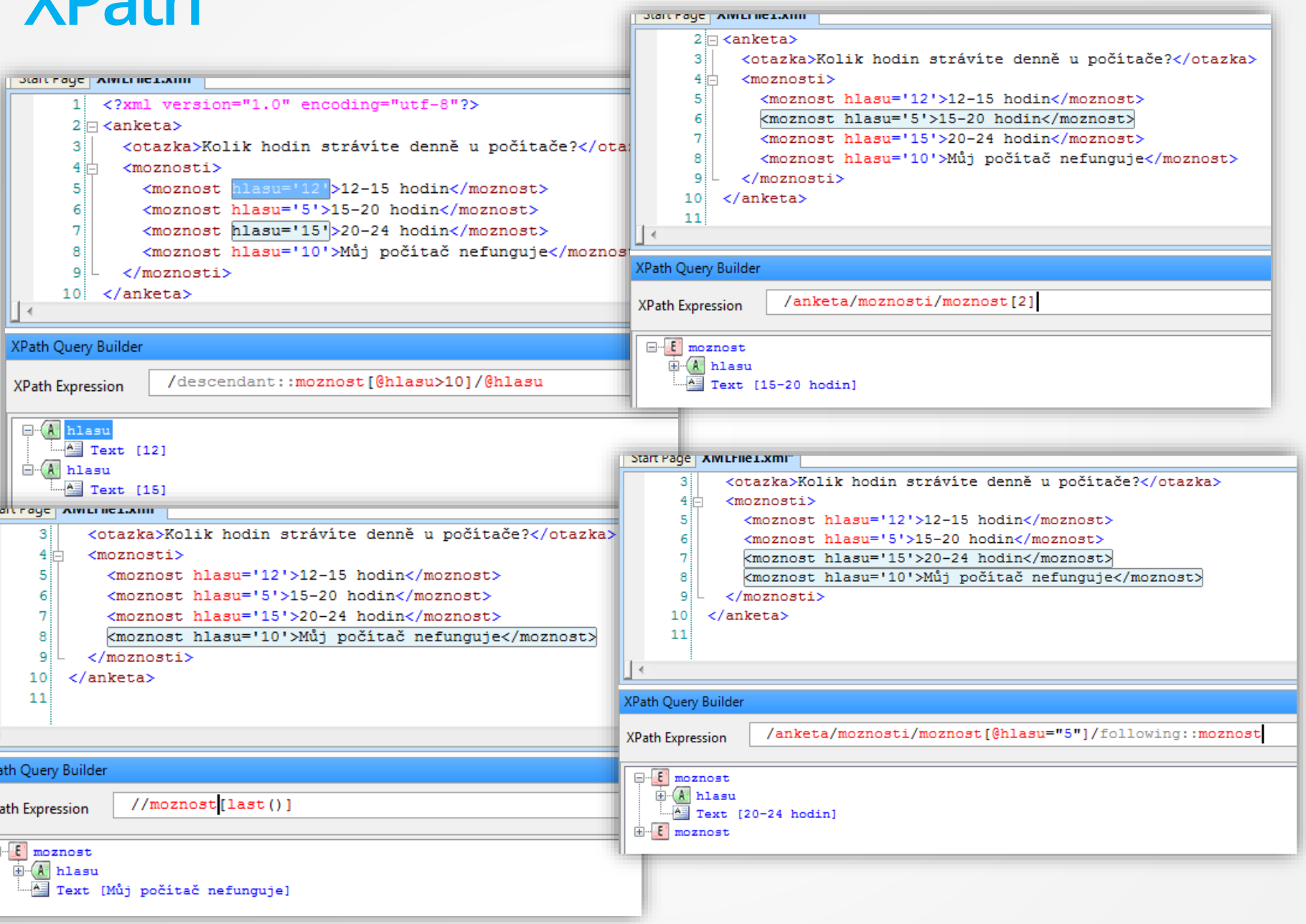
- Specifikace uzlu
  - názvem (včetně použití prefixu pro jmenný prostor)
  - typem (text(), node(), comment(), processing-instruction())



XPath – predikáty, atd.

- Je možné používat
  - Znaky „*“, „.“, „..“
  - Matematické, relační i logické operátory)
  - Zkratkový znak „@“ pro osu attribute::
  - Funkce (cca 100 funkcí) (last(), position(), string(), concat(), atd.)
- Podmínky je možné konstruovat s ohledem na všechny prvky ve vztahu k danému elementu (tj. osy, testy uzlů i atributů)

XPath


1 <?xml version="1.0" encoding="utf-8"?>
2 <anketa>
3 <otazka>Kolik hodin strávíte denně u počítače?</ota
4 <moznosti>
5 <moznost hlasu='12'>12-15 hodin</moznost>
6 <moznost hlasu='5'>15-20 hodin</moznost>
7 <moznost hlasu='15'>20-24 hodin</moznost>
8 <moznost hlasu='10'>Můj počítač nefunguje</moznos
9 </moznosti>
10 </anketa>
XPath Query Builder
XPath Expression
/descendant::moznost[@hlasu>10]/@hlasu
hlasu
Text [12]
hlasu
Text [15]
2 <anketa>
3 <otazka>Kolik hodin strávíte denně u počítače?</otazka>
4 <moznosti>
5 <moznost hlasu='12'>12-15 hodin</moznost>
6 <moznost hlasu='5'>15-20 hodin</moznost>
7 <moznost hlasu='15'>20-24 hodin</moznost>
8 <moznost hlasu='10'>Můj počítač nefunguje</moznost>
9 </moznosti>
10 </anketa>
11
XPath Query Builder
XPath Expression
//moznost[last()]
moznost
hlasu
Text [Můj počítač nefunguje]
XPath Query Builder
XPath Expression
/anketa/moznosti/moznost[2]
moznost
hlasu
Text [15-20 hodin]
3 <otazka>Kolik hodin strávíte denně u počítače?</otazka>
4 <moznosti>
5 <moznost hlasu='12'>12-15 hodin</moznost>
6 <moznost hlasu='5'>15-20 hodin</moznost>
7 <moznost hlasu='15'>20-24 hodin</moznost>
8 <moznost hlasu='10'>Můj počítač nefunguje</moznost>
9 </moznosti>
10 </anketa>
11
XPath Query Builder
XPath Expression
/anketa/moznosti/moznost[@hlasu="5"]/following::moznost
moznost
hlasu
Text [20-24 hodin]
moznost

XPath

```
<otazka>Kolik hodin strávíte denně u počítače?</otazka>
<moznosti>
  <moznost hlasu='12'>12-15 hodin</moznost>
  <moznost hlasu='5'>15-20 hodin</moznost>
  <moznost hlasu='15'>20-24 hodin</moznost>
  <moznost hlasu='10'>Můj počítač nefunguje</moznost>
</moznosti>
</anketa>
```

XPath Query Builder

XPath Expression: `//moznost[starts-with(.,'M')]`

- moznost
  - hlasu
  - Text [Můj počítač nefunguje]

```
<?xml version="1.0" encoding="utf-8"?>
<anketa>
  <otazka>Kolik hodin strávíte denně u počítače?</otazka>
  <moznosti>
    <moznost hlasu='12'>12-15 hodin</moznost>
    <moznost hlasu='5'>15-20 hodin</moznost>
    <moznost hlasu='15'>20-24 hodin</moznost>
    <moznost hlasu='10'>Můj počítač nefunguje</moznost>
  </moznosti>
</anketa>
```

XPath Query Builder

XPath Expression: `//*[string-length(text())>20]`

- otazka
  - Text [Kolik hodin strávíte denně u počítače?]
- moznost

```
<?xml version="1.0" encoding="utf-8"?>
<anketa>
  <otazka>Kolik hodin strávíte denně u počítače?</otazka>
  <moznosti>
    <moznost hlasu='12'>12-15 hodin</moznost>
    <moznost hlasu='5'>15-20 hodin</moznost>
    <moznost hlasu='15'>20-24 hodin</moznost>
    <moznost hlasu='10'>Můj počítač nefunguje</moznost>
  </moznosti>
</anketa>
```

XPath Query Builder

XPath Expression: `//moznosti[moznost='20-24 hodin']/moznost`

- moznost
  - hlasu
  - Text [12-15 hodin]
- moznost
- moznost
- moznost

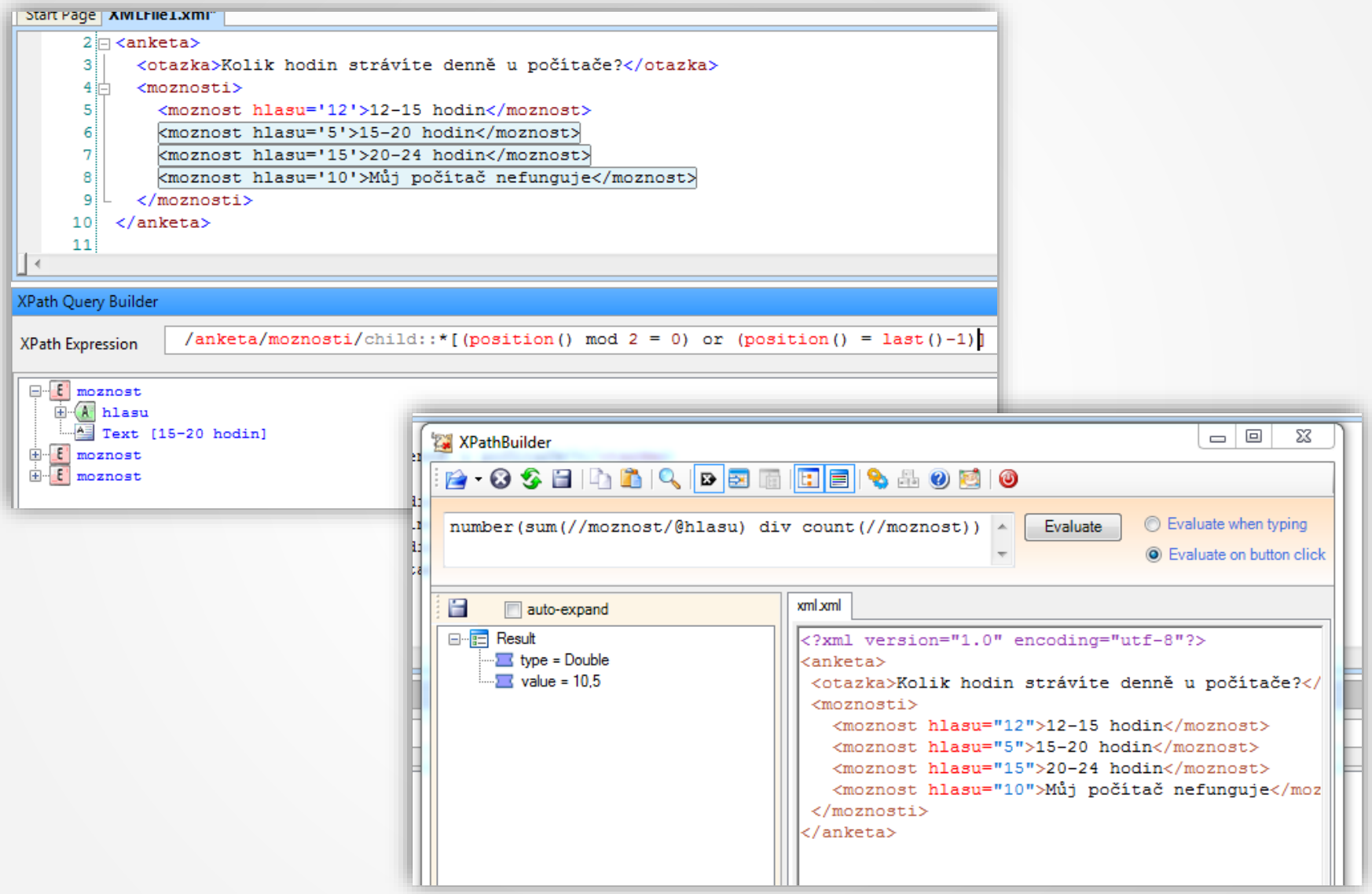
```
<?xml version="1.0" encoding="utf-8"?>
<anketa>
  <otazka>Kolik hodin strávíte denně u počítače?</otazka>
  <moznosti>
    <moznost hlasu='12'>12-15 hodin</moznost>
    <moznost hlasu='5'>15-20 hodin</moznost>
    <moznost hlasu='15'>20-24 hodin</moznost>
    <moznost hlasu='10'>Můj počítač nefunguje</moznost>
  </moznosti>
</anketa>
```

XPath Query Builder

XPath Expression: `//*[count(child::*)>3]`

- moznosti
  - moznost
  - moznost
  - moznost
  - moznost

XPath


Start Page
XMLFile1.xml
<anketa>
<otazka>Kolik hodin strávíte denně u počítače?</otazka>
<moznosti>
<moznost hlasu='12'>12-15 hodin</moznost>
<moznost hlasu='5'>15-20 hodin</moznost>
<moznost hlasu='15'>20-24 hodin</moznost>
<moznost hlasu='10'>Můj počítač nefunguje</moznost>
</moznosti>
</anketa>
XPath Query Builder
XPath Expression
/anketa/moznosti/child::*[(position() mod 2 = 0) or (position() = last()-1)]
moznost
hlasu
Text [15-20 hodin]
moznost
moznost
XPathBuilder
number(sum(//moznost/@hlasu) div count(//moznost))
Evaluate
Evaluate when typing
Evaluate on button click
auto-expand
Result
type = Double
value = 10,5
xml.xml
<?xml version="1.0" encoding="utf-8"?>
<anketa>
<otazka>Kolik hodin strávíte denně u počítače?</
<moznosti>
<moznost hlasu="12">12-15 hodin</moznost>
<moznost hlasu="5">15-20 hodin</moznost>
<moznost hlasu="15">20-24 hodin</moznost>
<moznost hlasu="10">Můj počítač nefunguje</moz
</moznosti>
</anketa>

XPATH a JavaScript

- Využité metodu evaluate na objektu s DOMem
- Umí pracovat také s namespace prostřednictvím tzv. Resolverů
- https://developer.mozilla.org/en-US/docs/Web/XPath/Introduction_to_using_XPath_in_JavaScript

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        showResult(xhttp.responseXML);
    }
};
xhttp.open("GET", "books.xml", true);
xhttp.send();

function showResult(xml) {
    var txt = "";
    path = "/bookstore/book/title"
    if (xml.evaluate) {
        var nodes = xml.evaluate(path, xml, null, XPathResult.ANY_TYPE, null);
        var result = nodes.iterateNext();
        while (result) {
            txt += result.childNodes[0].nodeValue + "<br>";
            result = nodes.iterateNext();
        }
     }
    document.getElementById("demo").innerHTML = txt;
}
```

JSON

- JavaScript Object Notation
  - Kolekce párů název/hodnota
  - Seznam hodnot
  - Datové typy – JSONString, JSONNumber, JSONBoolean, JSONNull, atd.
- Vhodný pro výměnu a přenos krátkých strukturovaných dat
- Možné využívat i JSON Schema pro validaci (https://json-schema.org)
- Pozor na datum a čas – string podle normy ISO 8601

- JSON.parse() vs. JSON.stringify()

- http://jsonlint.com/

JSON

```
{
  "@context": "http://schema.org",
  "@type": "ItemList",
  "name": "Seznam produktů",
  "itemListElement": [
    {
      "@type": "Product",
      "name": "Kvalitní boty",
      "description": "Elegantní boty pro každou příležitost.",
      "offers": {
        "@type": "Offer",
        "price": "49.99",
        "priceCurrency": "USD",
        "availability": "http://schema.org/InStock"
      }
    },
    {
      "@type": "Product",
      "name": "Moderní tričko",
      "description": "Stylové tričko s moderním designem.",
      "offers": {
        "@type": "Offer",
        "price": "29.99",
        "priceCurrency": "USD",
        "availability": "http://schema.org/OutOfStock"
      }
    ],
  "datePublished": "2023-10-29T15:30:00"
}
```

JSON a JavaScript

```
function loadJSON()
{
   var data_file = "http://www.tutorialspoint.com/json/data.json";
   var http_request = new XMLHttpRequest();

   http_request.onreadystatechange  = function(){
      if (http_request.readyState == 4  )
      {
        // Javascript function JSON.parse to parse JSON data
        var jsonObj = JSON.parse(http_request.responseText);

        // jsonObj variable now contains the data structure and can
        // be accessed as jsonObj.name and jsonObj.country.
        document.getElementById("Name").innerHTML =  jsonObj.name;
        document.getElementById("Country").innerHTML = jsonObj.country;
      }
   }
   http_request.open("GET", data_file, true);
   http_request.send();
}
```

```
// URL k JSON souboru na serveru
var url = 'https://example.com/products.json';

// Načteme JSON data z externího souboru
fetch(url)
  .then(response => response.json())
  .then(products => {

   products.forEach(product => {
      console.log('Název produktu: ' + product.name);
      console.log('Cena produktu: ' + product.price);
      console.log('--------------------------');
    });
  })
  .catch(error => {
    console.error('Chyba při načítání dat: ' + error);
  });
```