Development of Internet Applications

AJAX, JSON, XML

Ing. Michal Radecký, Ph.D.

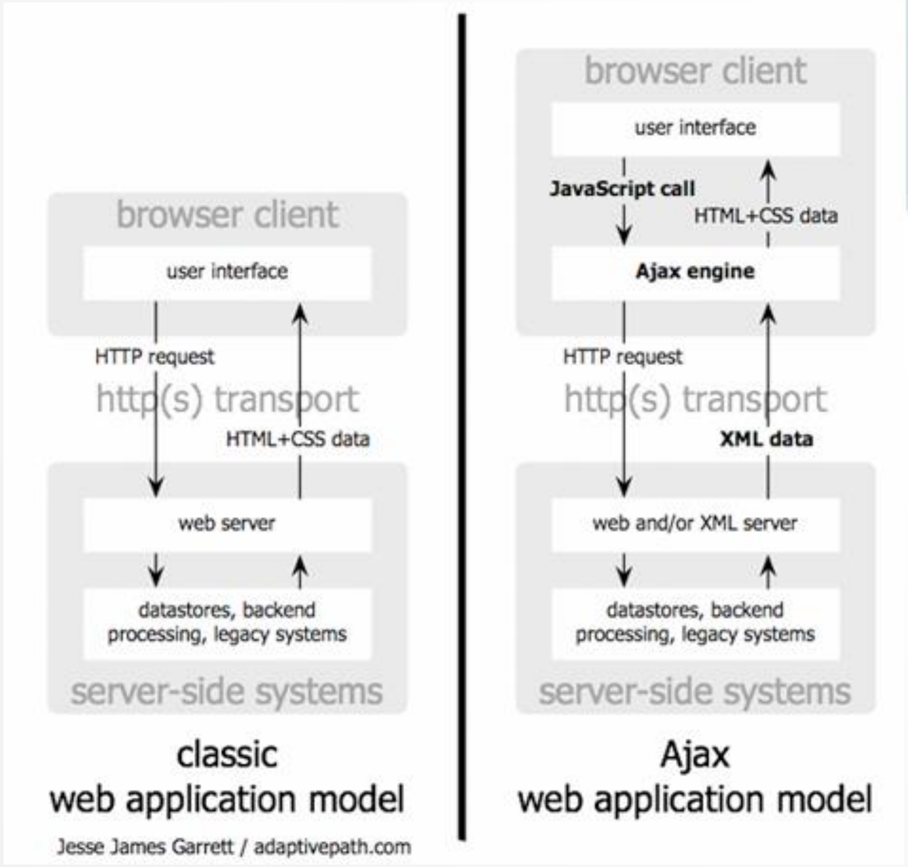
www.cs.vsb.cz/radecky

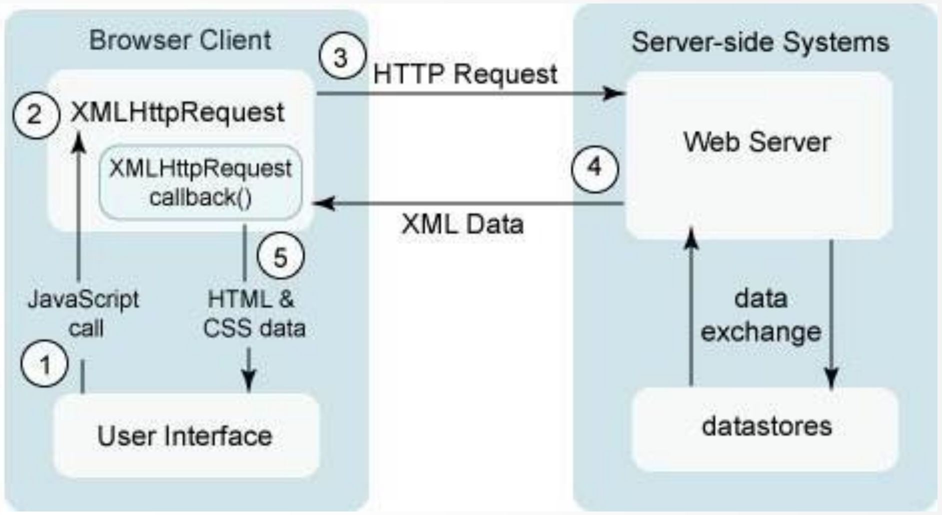
What is AJAX

- Asynchronous JavaScript and XML
- Combination of technologies that offer ability to change parts of web pages based on received data (HTTP requests and responses); without necessity of page reload.
- Based on history approaches (IFRAME, LAYER, Aplets, etc.), first mentioned in 2005 – in nowadays form
- Pros
    - Higher user experiences and efficiency of web applications usage
    - Lower demands on data amount
- Cons
    - Elimination of Back button (browser history)
    - Changes within the pages doesn't change page itself (URL)

Operational model

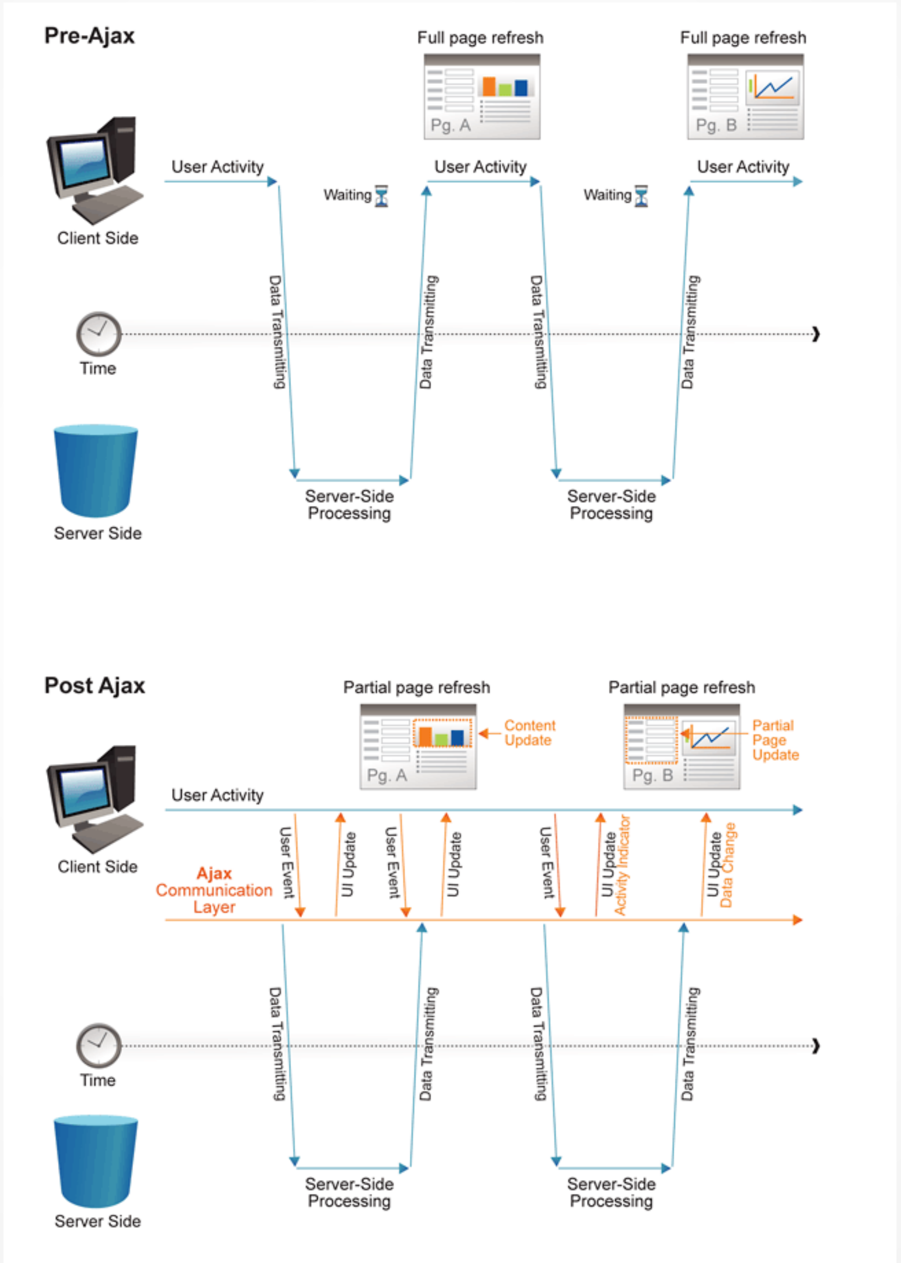
Zdroj: http://www.eioba.com





Operational model

Zdroj: http://www.websiteoptimization.com



AJAX and implementation

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

- DOM and XmlHttpRequest
- Possible usage of frameworks (not only Javascript, .NET, Java, Python, etc.)

```
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");
    }
 }
```

Object creation

```
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');
          }
      }
  }
```

AJAX call

AJAX and jQuery

Zdroj: http://www.ibm.com

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

Loading of HTML content

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

Sending data to server (POST)

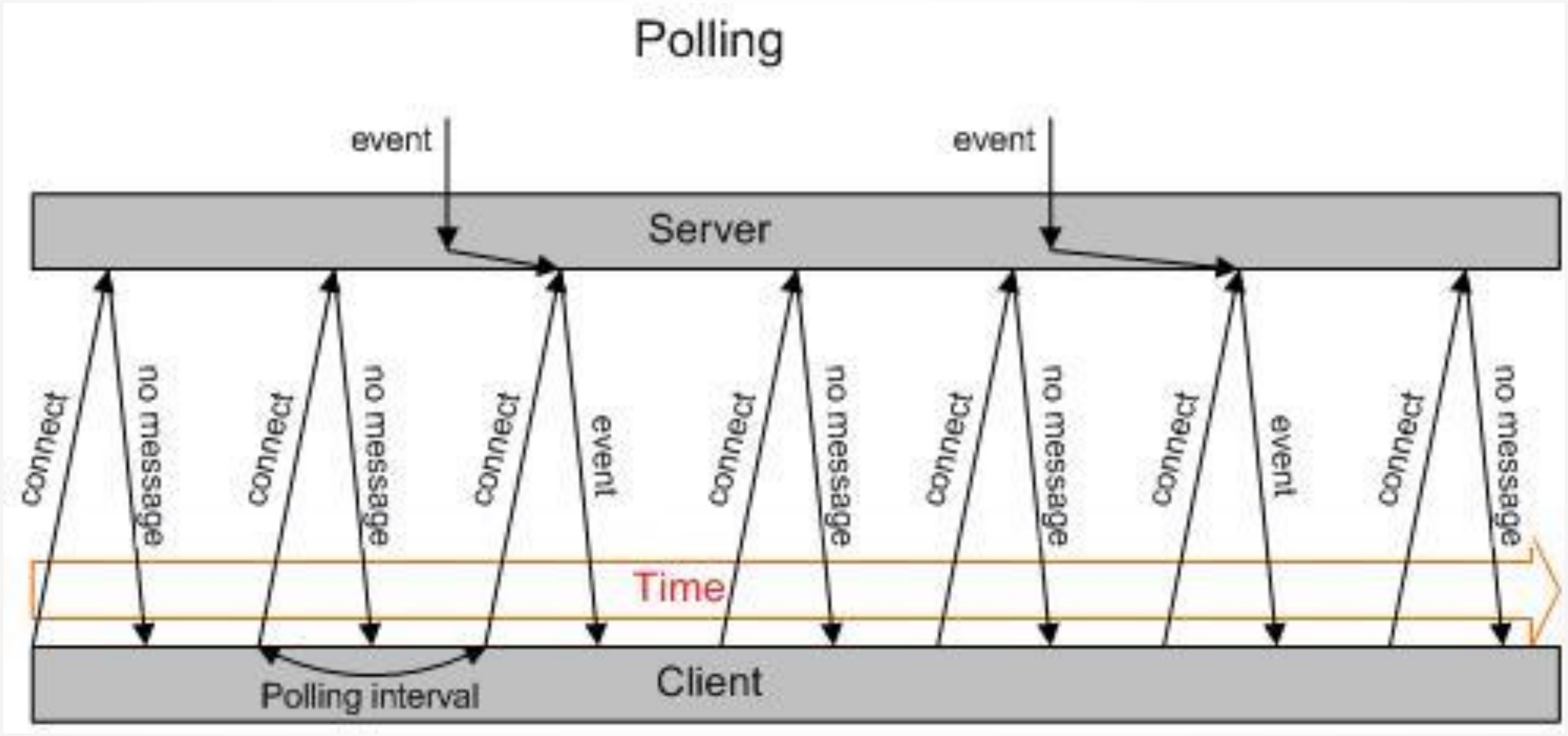
```
$.ajax({
    url: 'document.xml',
    type: 'GET',
    dataType: 'xml',
    timeout: 1000,
    error: function(){
        alert('Error loading XML document');
    },
    success: function(xml){
        $(xml).find('item').each(function(){
        var item_text = $(this).text();

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

Complex example of XML processing based on AJAX request

Asynchrony approaches

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

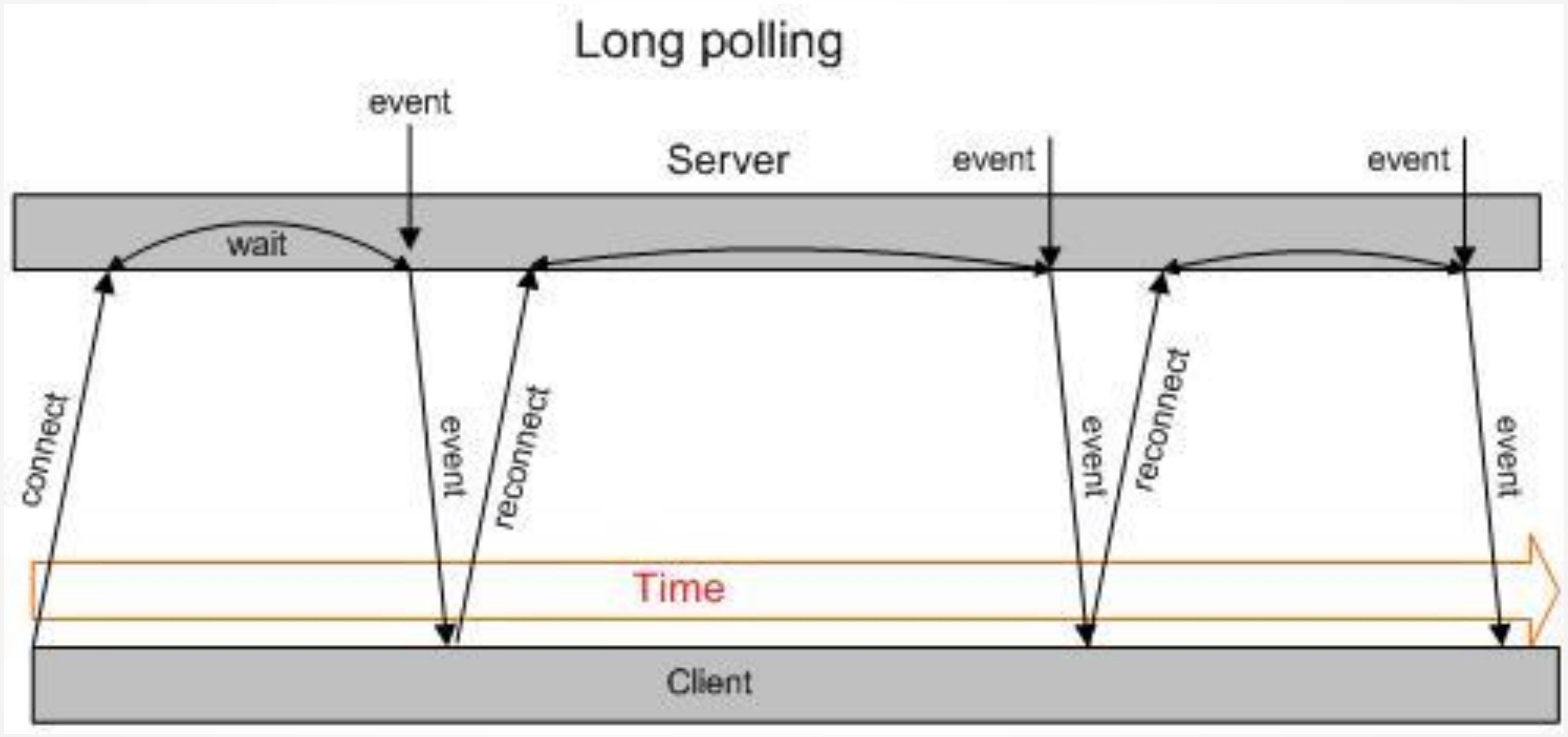
- Polling



Asynchrony approaches

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

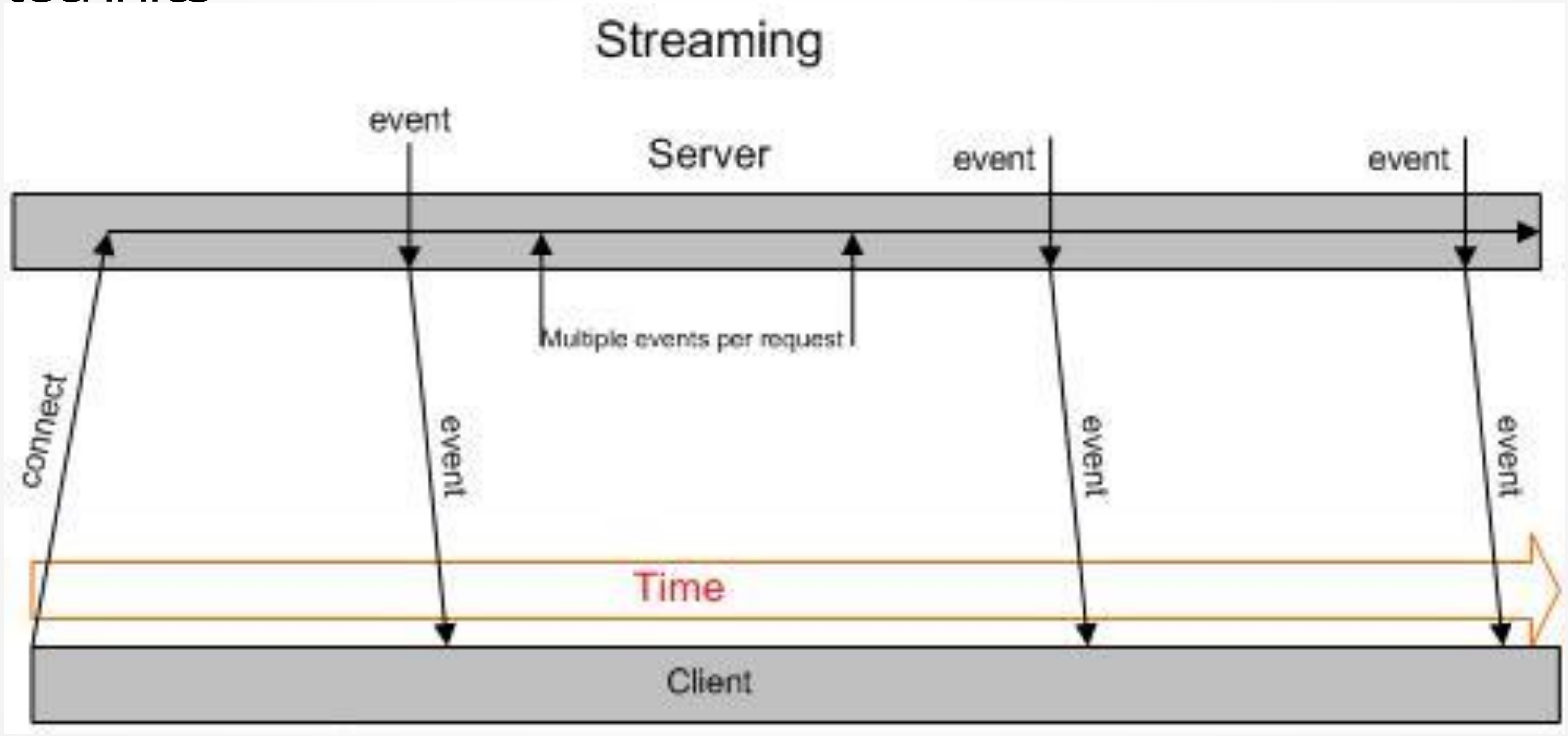
- Long - polling



Asynchrony approaches

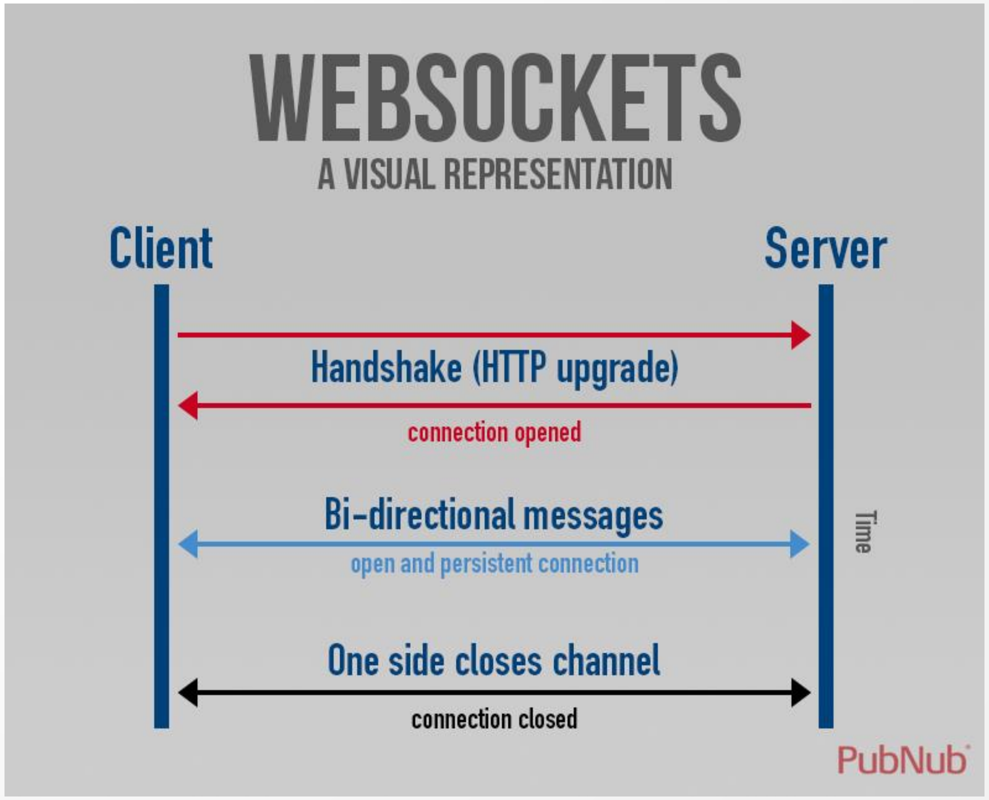
Zdroj: http://pic.dhe.ibm.com/infocenter

- Streaming
- Push aproach
- Comet, reverse AJAX – many implementations, different technics



WebSockets

- Persistent two-way communication channel
- Based on WebSocket object
- send, onmessage, onopen, onerror, readyState



What is XML

- eXtensible Markup Language
- A set of rules
  - Semantic markup (tags, elements)
  - Structure of document
  - Identification of parts of the document
- Language for describing other languages
  - meta-markup language
  - Define syntax of another language (XML based)
- Based on SGML (Standard Generalized Markup Language)
  - Same features
  - Simplicity
- It is not another markup language
  - meta-language
  - Particular names of elements, attributes, etc. is up to developer

Why use XML

- Data + markup = structured data with semantics
- Enables specification of relations between elements
- It can be 100% ASCII text
- It has detailed specification by W3C
- No patent, no copyright and other restrictions
- There is no version of XML (itself)
- Huge support in many programming languages
- Support in development tools
- Easy processing

XML format

jsou rozšiřitelné
drží strukturu dokumentu

- Elements/Tags
  - Markup defines XML structure beside text content
  - Markup is almost tags/elements
    - tag is everything what begins '<' and ends '>'
    - tag has a name
      - Begins with **[a-z,A-Z,_]**
      - Case-sensitive (**<B>** vs. **<b>**)
  - Empty tag
    - No content, can have atributes
    - Simple syntax based on '**/>**'

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

- Entities

```
<tag attribute=„value">
  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>
```

XML format

- Attributes
  - Included within beginning elements and empty elements
  - Couple `jméno = hodnota`
  - Name
    - begins `[a-z,A-Z,_]`
    - Only one attribute with same name within one element
  - Value
    - *string* in quotes
    - Any characters
    - Quotes rule – no crossing

Information about document without relation to document

Possibility to add information without changes of document structure

Data location

- data of XML document can be located
  - In attributes
  - In content of elements
- recommendations
  - Data itself (main data) within elements
  - Information on data (meta-data) in attributes
  - In attributes usually
    - ID numbers
    - URL
    - information with low value or priority for readers

```
<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>
  ...
```

Other specifications

- Comments
  - “`<!--`”…“`-->`”
- Text without interpretation
  - section **`CDATA`**

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

- Instructions of other aplication
  - “`<?nazev `”…“`?>`”

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

- XML Prolog

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

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

Namespace

- Namespace
  - Separation of different sets of specified elements based on prefix
  - Specification and usage based on **`xmlns:název`**
  - Validity for descendants
  - NS specification is related to URI (can exists or not)

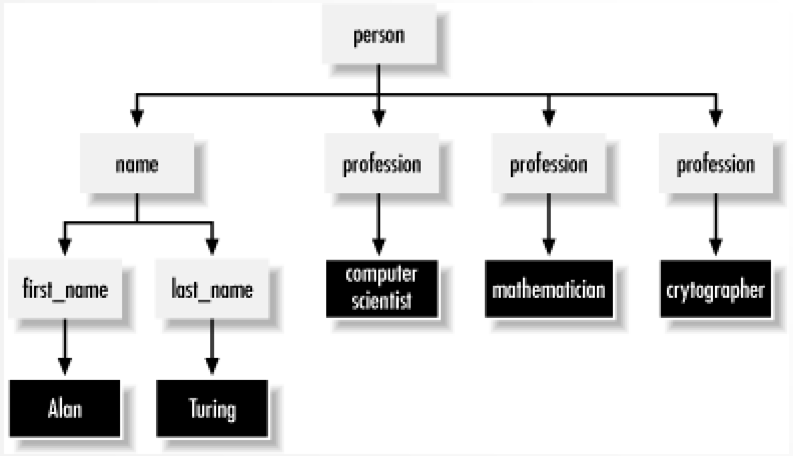
```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="keyword">
  ...
  </xsl:template>
</xsl:stylesheet>
```

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

Parent, childs, ...

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

- XML documents equals to tree structure
- Only one root element is allowed
- No crossing rule
- There is parent of each element and childs of each element (parent is max. one, childs can be 0 or more)



```
<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
- Language for describing rules and possibilites of XML document creation
- Used for validation of XML document
- Defines
    - List of elements, attributes, notations and entities
    - Content of elements and attributes
    - Relations betwen them
    - Structure
- Location
    - In prolog after declaration
    - Before first element
- Directly DTD syntax or URL targeted DTD file

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

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

DTD – element declarations

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

- ANY
  - Any content of element is allowed (child elements or #PCDATA)
- EMPTY
  - Element without content
- (#PCDATA)
  - Parsed character data
- (child1, child2, …)
  - Declaration of list of childs
  - Regular definitions of multiplicity can be used (child1?, child2+, child3*)
- (child1 | child2)
  - OR choice
- Usage of brackets for complex specifications

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

DTD – attribute declaration

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

- CDATA
  - Parsed text
- NMTOKEN, NMTOKENS
  - Value based on name specification, e.g. name in HTML
- (monday|tuesday|wednesday)
  - A set of possible values
- ID
  - unique identification insdie document
- IDREF, IDREFS
  - Relation to element with ID attribute
- ENTITY, ENTITIES
  - Link to defined entity

- „value“
  - Particular value
- #IMPLIED
  - Attribute is optional
- #REQUIRED
  - Attribut eis required
- #FIXED “value”
  - If attribute is mentioned, has to have this value

DTD – entity declaration

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

- „value“
  - Particular value
- SYSTEM „external source url“

```
<!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 and XML

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 developm
   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)

- Cons of DTD
  - No support for namespaces
  - Unable to specify data types
  - DTD syntax is not based on XML
- XML Schema
  - Specification language based on XML
  - W3C recommendation
  - Defines
    - Structure of XML document
    - Elements and attributes of XML document
    - Child elements, their number and order
    - Content of element
    - Data types of element and attributes (more than 40 types)
    - Default and fixed values
  - Support for namespaces (NS xs: for XML Schema)

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 – element declaration

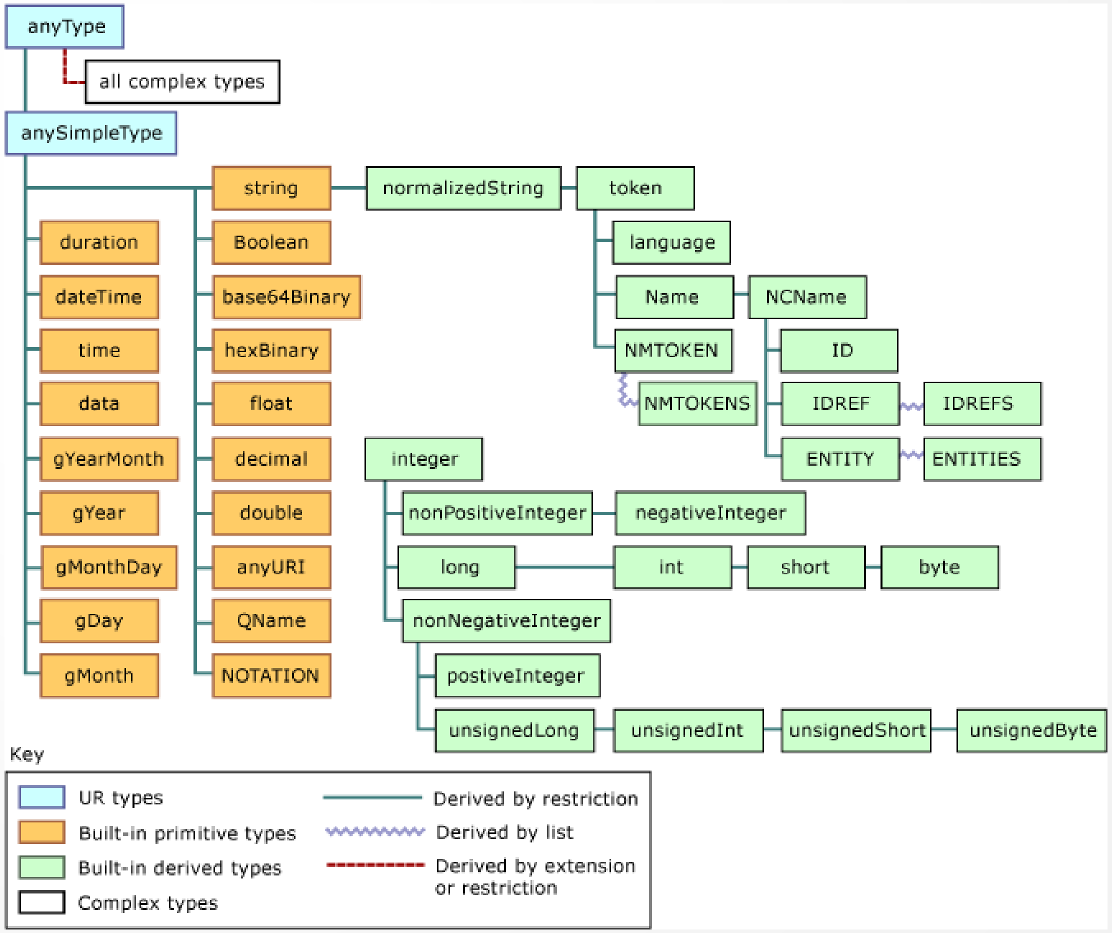
```
<xs:element name=„name" type=„type"/>
```

simple element

- Name based on standard rules
- Type from defined set of standard types or possibility of custom data types

```
<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 – attribute declaration

- Each attribute is specified as simple-element as a art of complex-element

complex element

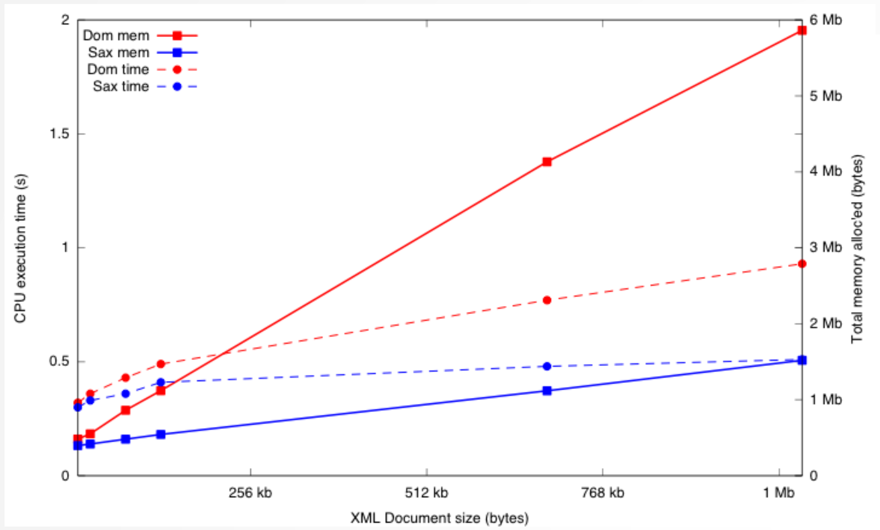
```
<xs:element name=„name“>
   <xs:complexType>
      <xs:sequence>
              <xs:element …/>
      </xs:sequence>
      <xs:attribute name=„name" type=„type"
                                 use="required"/>
   </xs:complexType>
</xs:element>
```

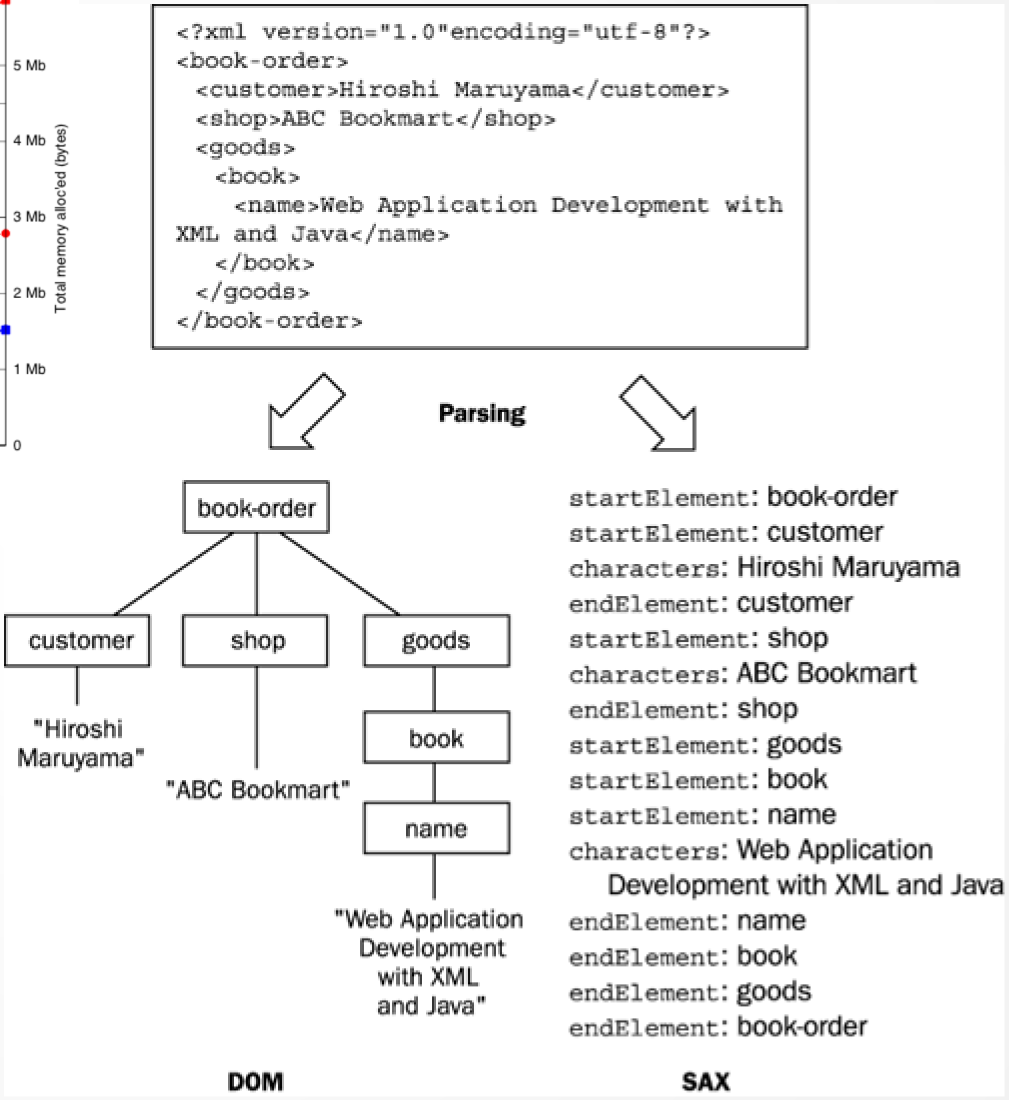
XML interface API

- DOM
    - Document Object Model
    - Tree structure of XML document based on object representation in memory
    - It is standard interface for XML access covered by W3C
    - higher demands on time and memory
- SAX
    - Simple API for XML – event-driven model
    - Processing of XML during its reading
    - Mathod calling – processing data at the beginning/ending of some element, text content, etc.
    - Fast, higher demands on implementation
- Parser in general
    - Application, software, class, algorithm
    - Its task is to proces XML document in text form and its transformation to another form for following utilization (eg. DOM)
    - Syntax checking, validation, DTD/XMLScheme specification

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>
```

```
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
```

JavaScript

Zdroj: http://www.interval.cz

- From XML to DOM

```
function verifyfunc() {
    if (xmlDoc.readyState != 4) {
        return false;
    }
}
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async="false";
xmlDoc.onreadystatechange=verifyfunc;
xmlDoc.load('xmltest.xml');
var xmlObj=xmlDoc.documentElement;
```

- Work with DOM

```
function WriteXML() {
    var t= "Otec: " + xmlObj.childNodes(0).text + " (narozen " +
xmlObj.childNodes(0).getAttribute("roknar") + ")\n"
    t += "Matka: " + xmlObj.childNodes(1).text + " (narozena " +
xmlObj.childNodes(1).getAttribute("roknar") + ")\n\n"
    t += "Děti:\n"
    var i;
    for(i=0; i<xmlObj.childNodes(2).childNodes.length; i++ ) {
        t += " " + xmlObj.childNodes(2).childNodes(i).text + " (narozen " +
xmlObj.childNodes(2).childNodes(i).getAttribute("roknar") + ")\n"
    }
    alert(t);
}
```

XPath

- The path (Path Expression) is main element for building queries
- Similar to path specification in file system
- Sequence of steps separated by „/“ or „//“
- Joining multiple sequences by OR - „|“
- Each step is formed by
  - Identification of axes
  - Node test (required)
  - Predicate
- The path is computed from left to right, relatively to current node

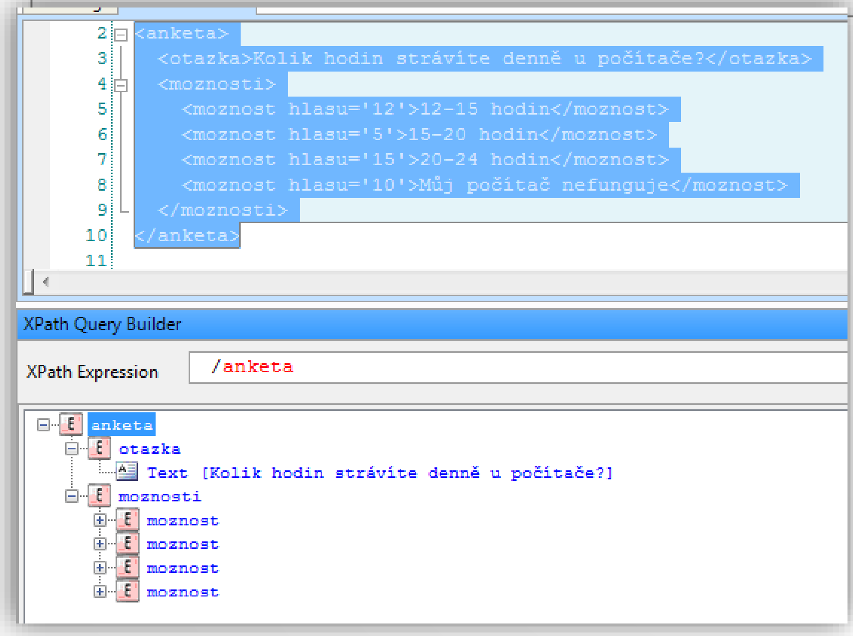
```
axisname::nodetest[predicate]
```

XPath – steps separation

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

Source XML file

```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>
```

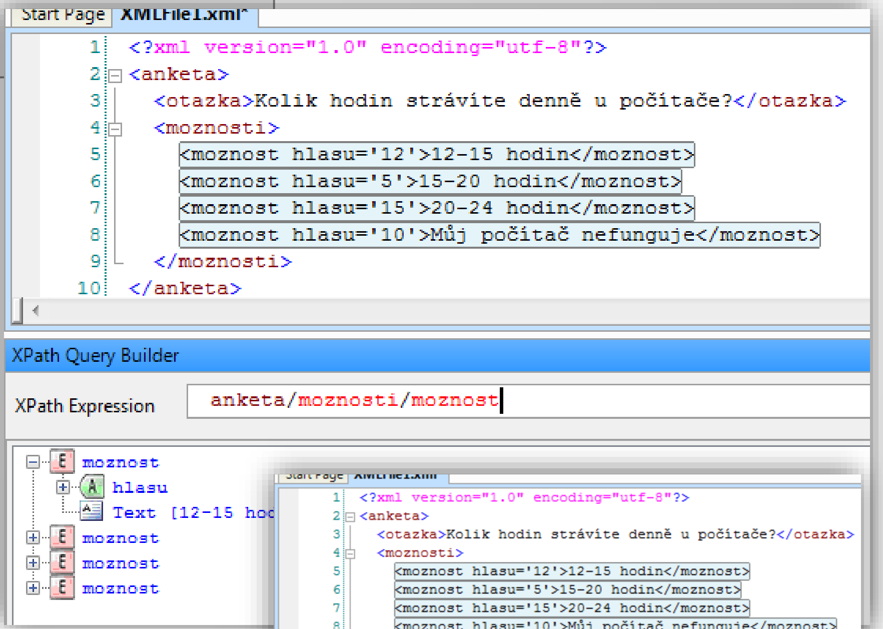


```xml
<anketa>
  <otazka>Kolik hodin strávíte denně u počítače?</otazka>
  <moznosti>
    <moznost hlasu='12'>12-15 hodin</moznost>
    <moznost hlasu='5'>15-20 hodin</moznost>
    <moznost hlasu='15'>20-24 hodin</moznost>
    <moznost hlasu='10'>Můj počítač nefunguje</moznost>
  </moznosti>
</anketa>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

XPath Query Builder

XPath Expression: `/anketa`

```
anketa
  otazka
    Text [Kolik hodin strávíte denně u počítače?]
  moznosti
    moznost
    moznost
    moznost
    moznost
```



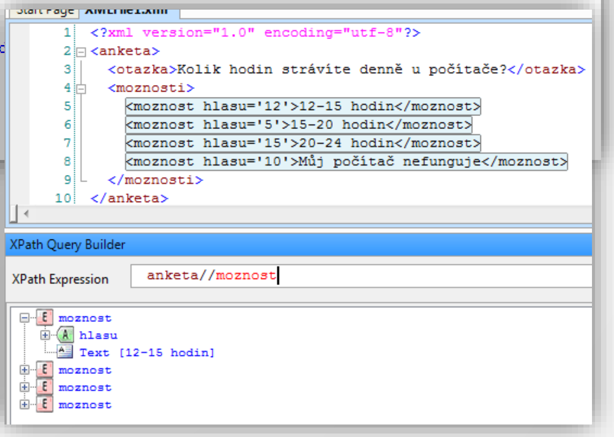
Start Page | XMLFile1.xml*

```xml
<?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: `anketa/moznosti/moznost`

```
moznost
  hlasu
  Text [12-15 hoc
moznost
moznost
moznost
```



Start Page | XMLFile1.xml

```xml
<?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

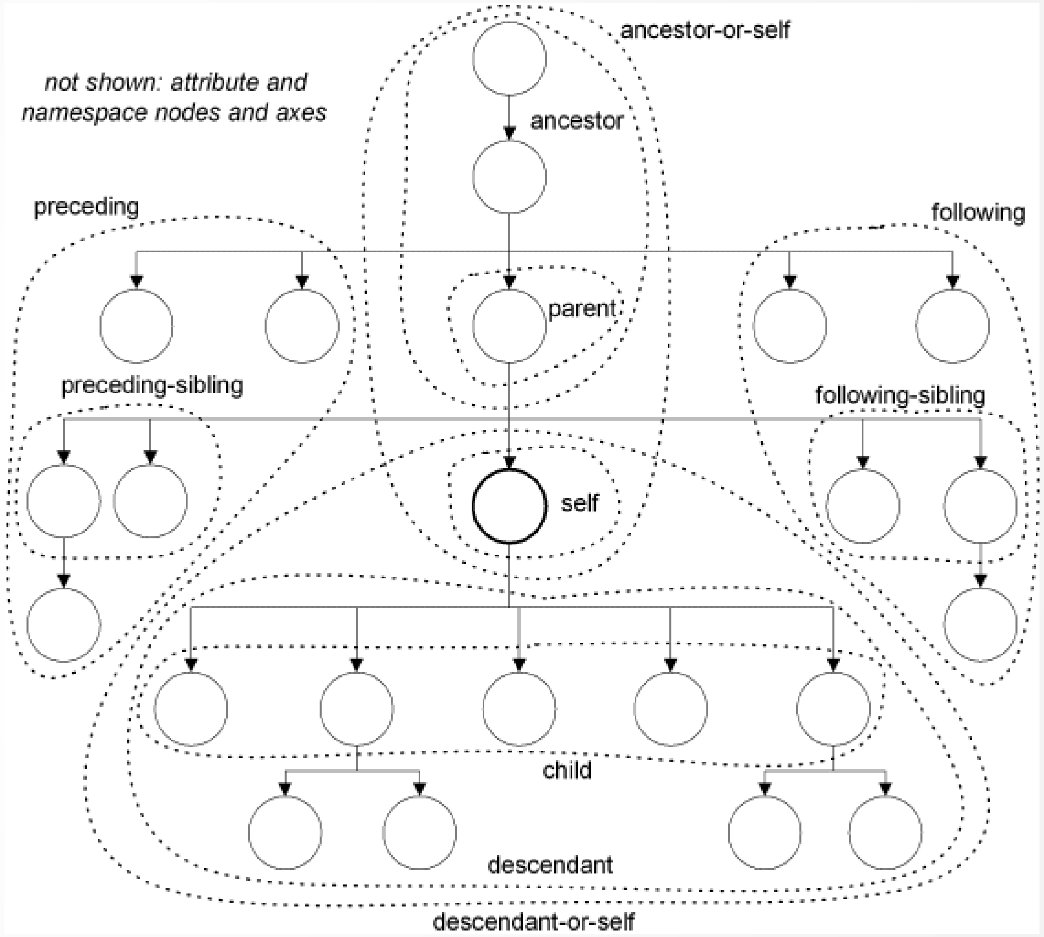
XPath Expression: `anketa//moznost`

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

XPath - Axes

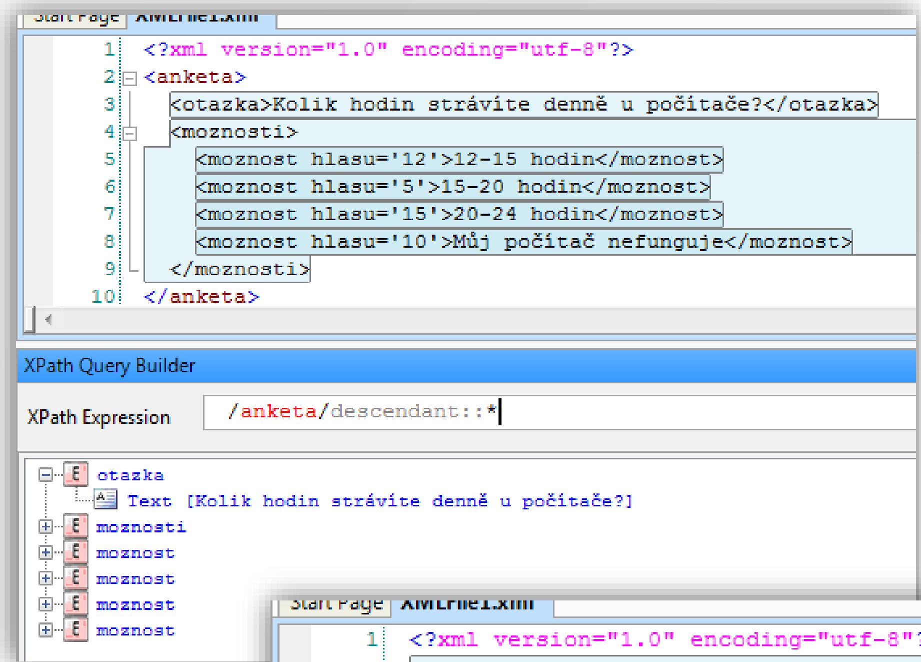
Zdroj: http://www.georgehernandez.com

- Define direction of XML tree quering
- Define a set of relevant nodes which are tested (evaluated), default (not specified) is axis: child::
- Axes *ancestor*, *descendant*, *following*, *preceding* and *self* are not overlap and they cover all nodes together



XPath - Axes

Zdroj: http://www.georgehernandez.com

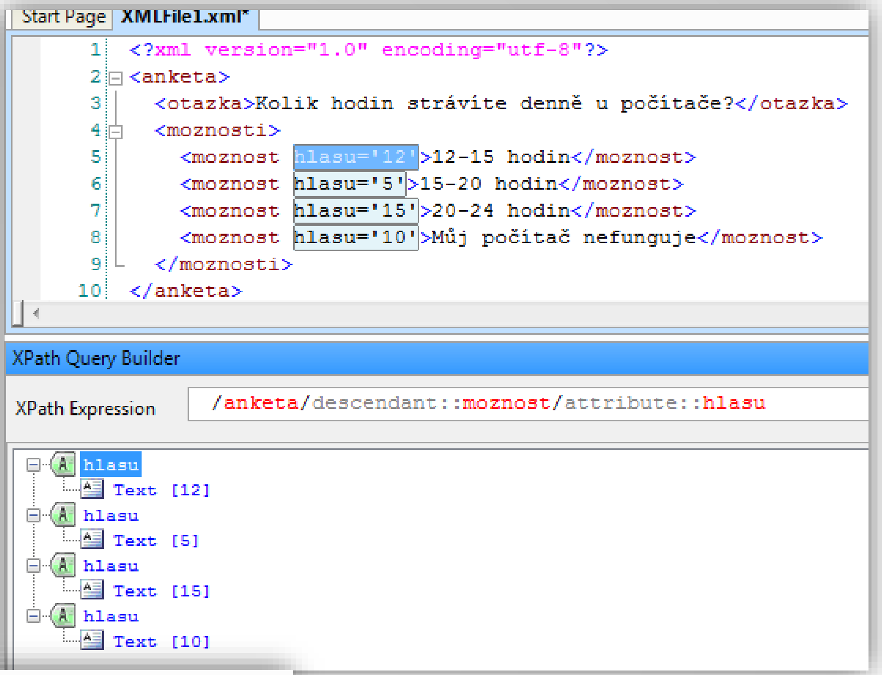


```
<?xml version="1.0" encoding="utf-8"?>
<anketa>
  <otazka>Kolik hodin strávite denně u počítače?</otazka>
  <moznosti>
    <moznost hlasu='12'>12-15 hodin</moznost>
    <moznost hlasu='5'>15-20 hodin</moznost>
    <moznost hlasu='15'>20-24 hodin</moznost>
    <moznost hlasu='10'>Můj počitač nefunguje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

XPath Query Builder

XPath Expression `/anketa/descendant::*`

```
otazka
  Text [Kolik hodin strávite denně u počítače?]
moznosti
moznost
moznost
moznost
moznost
```

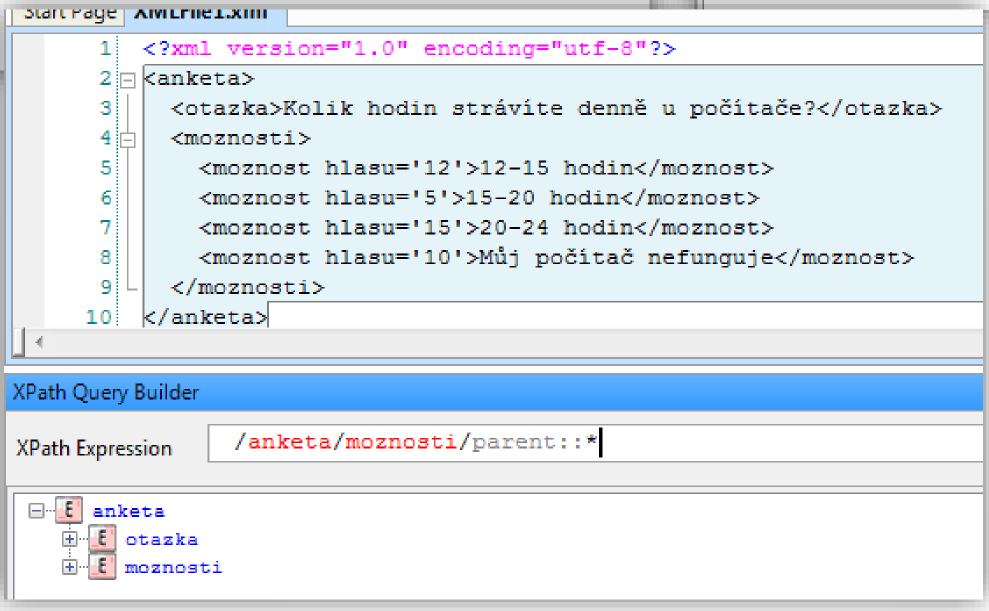


```
<?xml version="1.0" encoding="utf-8"?>
<anketa>
  <otazka>Kolik hodin strávite denně u počítače?</otazka>
  <moznosti>
    <moznost hlasu='12'>12-15 hodin</moznost>
    <moznost hlasu='5'>15-20 hodin</moznost>
    <moznost hlasu='15'>20-24 hodin</moznost>
    <moznost hlasu='10'>Můj počitač nefunguje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

XPath Query Builder

XPath Expression `/anketa/descendant::moznost/attribute::hlasu`

```
hlasu
  Text [12]
hlasu
  Text [5]
hlasu
  Text [15]
hlasu
  Text [10]
```



```
<?xml version="1.0" encoding="utf-8"?>
<anketa>
  <otazka>Kolik hodin strávite 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 počitač nefunguje</moznost>
  </moznosti>
</anketa>
```

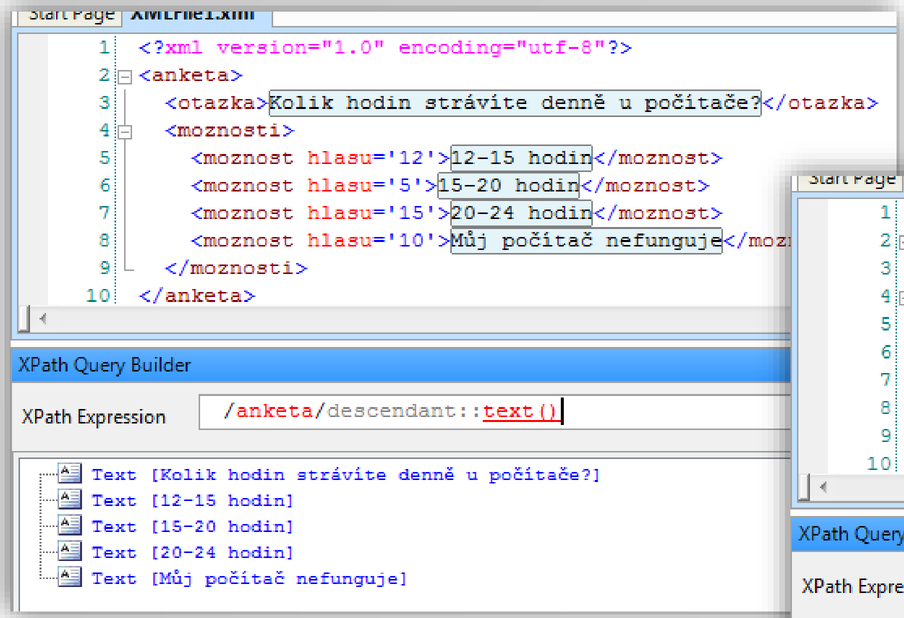
XPath Query Builder

XPath Expression `/anketa/moznosti/parent::*`

```
anketa
  otazka
  moznosti
```

XPath – Node test

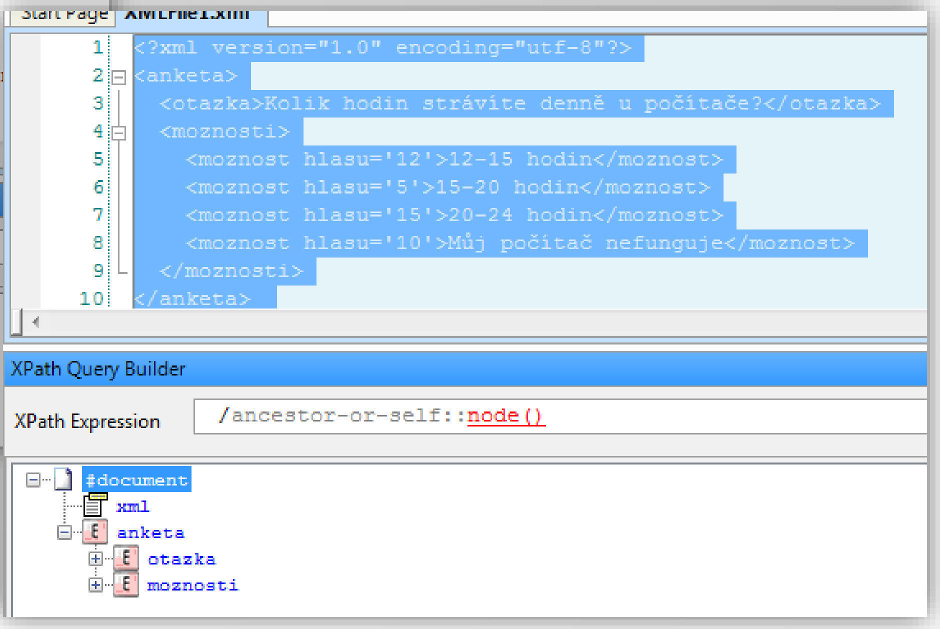
- Node specification
  - name (inc. Prefix for namespace)
  - type (text(), node(), comment(), processing-instruction())



```
<?xml version="1.0" encoding="utf-8"?>
<anketa>
  <otazka>Kolik hodin strávite 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</moz
  </moznosti>
</anketa>
```

```
/anketa/descendant::text()
```

```
Text [Kolik hodin strávite denně u počítače?]
Text [12-15 hodin]
Text [15-20 hodin]
Text [20-24 hodin]
Text [Můj počítač nefunguje]
```



```
<?xml version="1.0" encoding="utf-8"?>
<anketa>
  <otazka>Kolik hodin strávite 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>
```

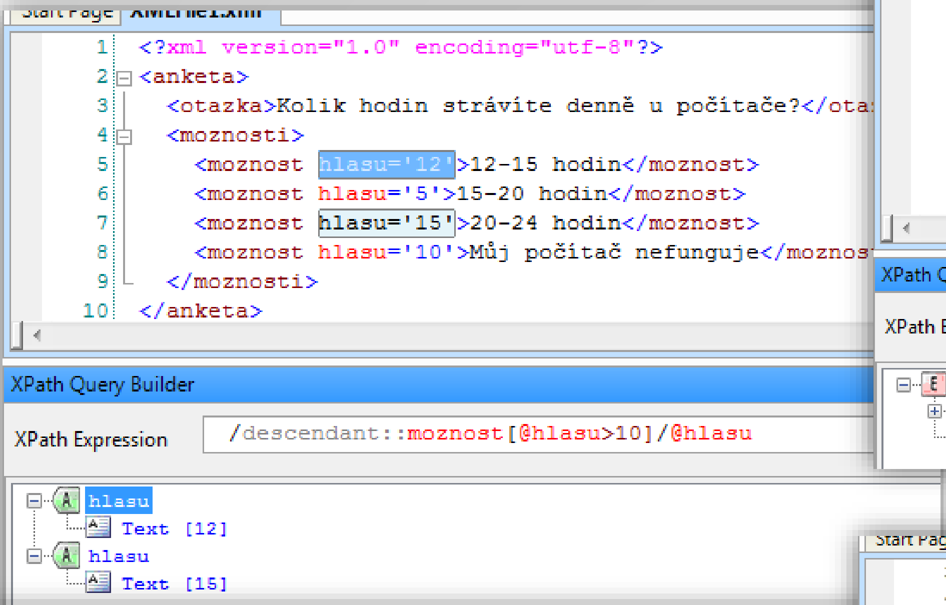
```
/ancestor-or-self::node()
```

```
#document
  xml
  anketa
    otazka
    moznosti
```

XPath – Predicates

- It is able to use
  - Characters „*“, „.“, „..“
  - Math, relation and logic operators
  - Substitution „@“ for attribute:: axis
  - Functions (100 funkcí) (last(), position(), string(), concat(), atd.)
- It is possible to define predicates in according to all elements related to a given element (axes, node test, attributes)

XPath



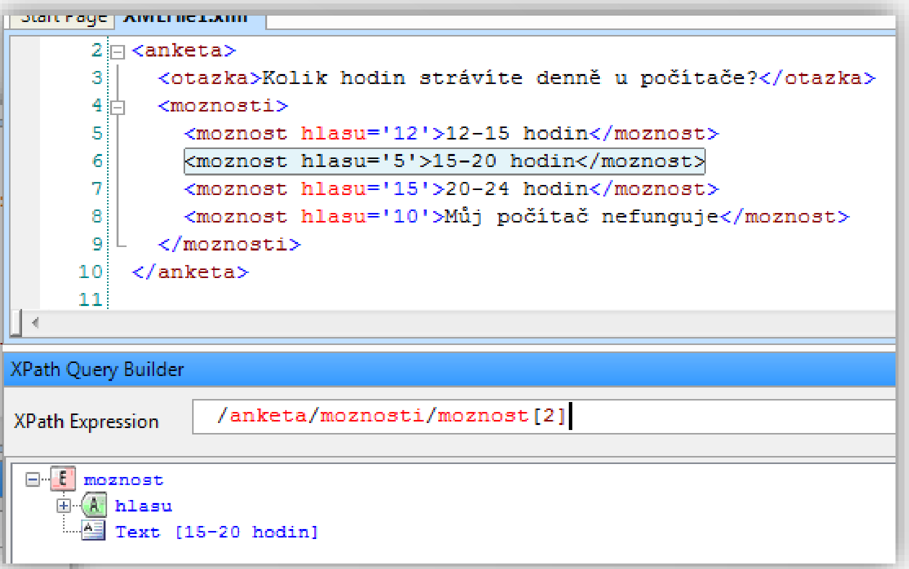
Start Page | XMLFile1.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<anketa>
  <otazka>Kolik hodin strávite denně u počítače?</ota
  <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</moznos
  </moznosti>
</anketa>
```

XPath Query Builder

XPath Expression `/descendant::moznost[@hlasu>10]/@hlasu`

```
hlasu
  Text [12]
hlasu
  Text [15]
```



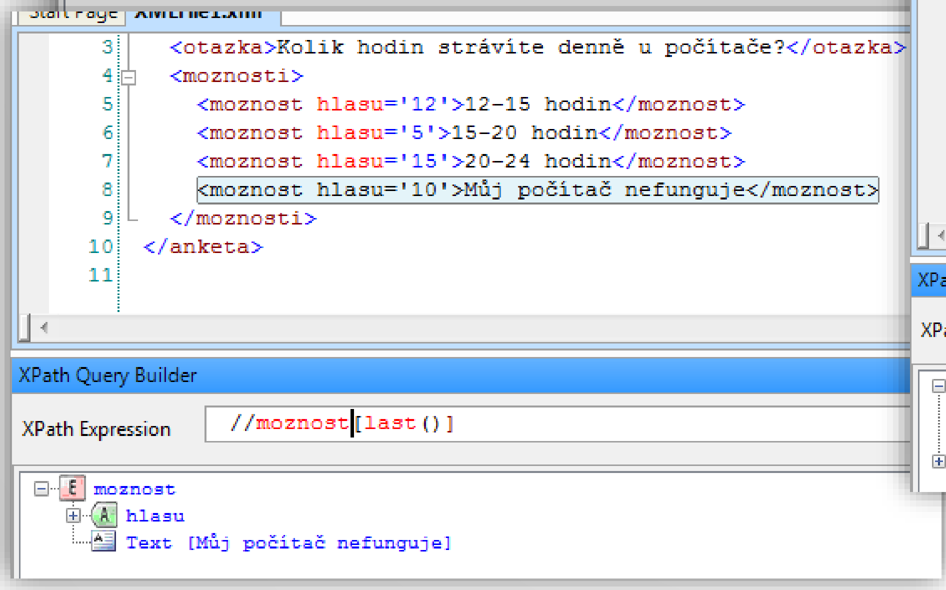
Start Page | XMLFile1.xml

```xml
<anketa>
  <otazka>Kolik hodin strávite 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/moznost[2]`

```
moznost
  hlasu
  Text [15-20 hodin]
```



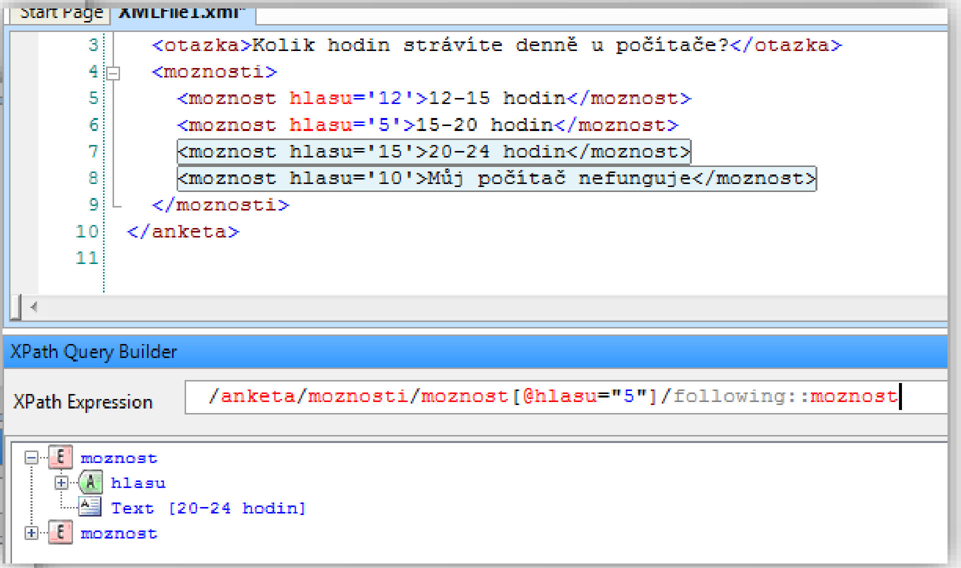
Start Page | XMLFile1.xml

```xml
  <otazka>Kolik hodin strávite 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[last()]`

```
moznost
  hlasu
  Text [Můj počítač nefunguje]
```



Start Page | XMLFile1.xml

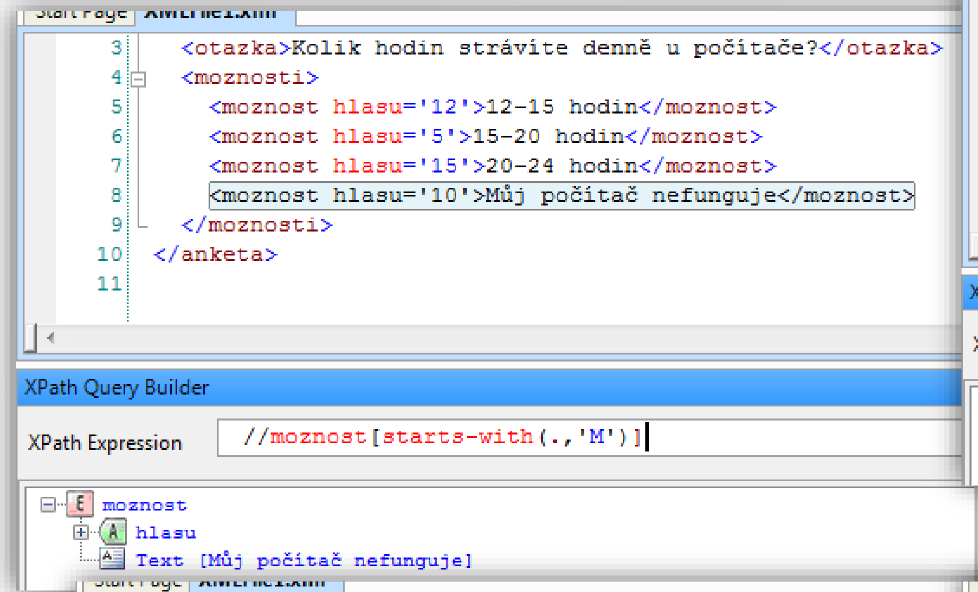
```xml
  <otazka>Kolik hodin strávite denně u počítače?</otazka>
  <moznosti>
    <moznost hlasu='12'>12-15 hodin</moznost>
    <moznost hlasu='5'>15-20 hodin</moznost>
    <moznost hlasu='15'>20-24 hodin</moznost>
    <moznost hlasu='10'>Můj počítač nefunguje</moznost>
  </moznosti>
</anketa>
```

XPath Query Builder

XPath Expression `/anketa/moznosti/moznost[@hlasu="5"]/following::moznost`

```
moznost
  hlasu
  Text [20-24 hodin]
moznost
```

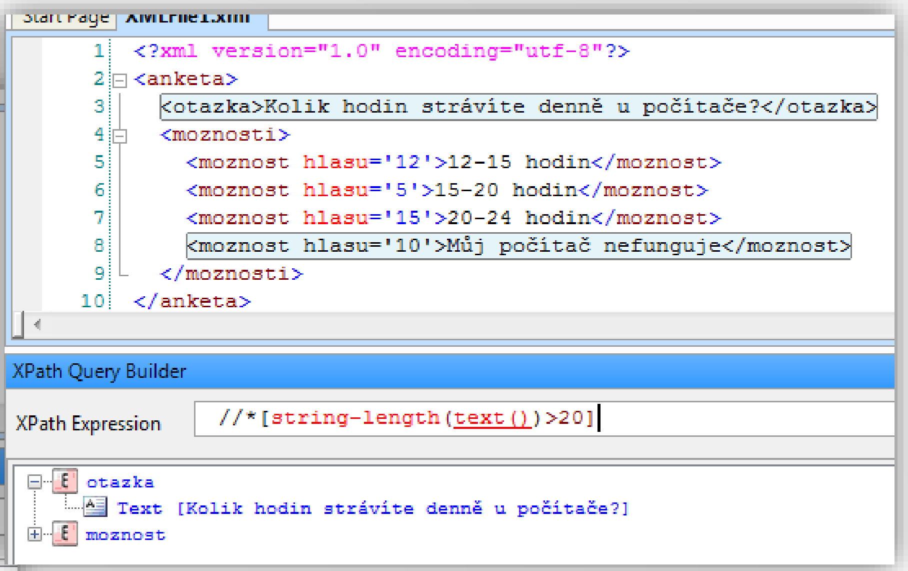
XPath



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

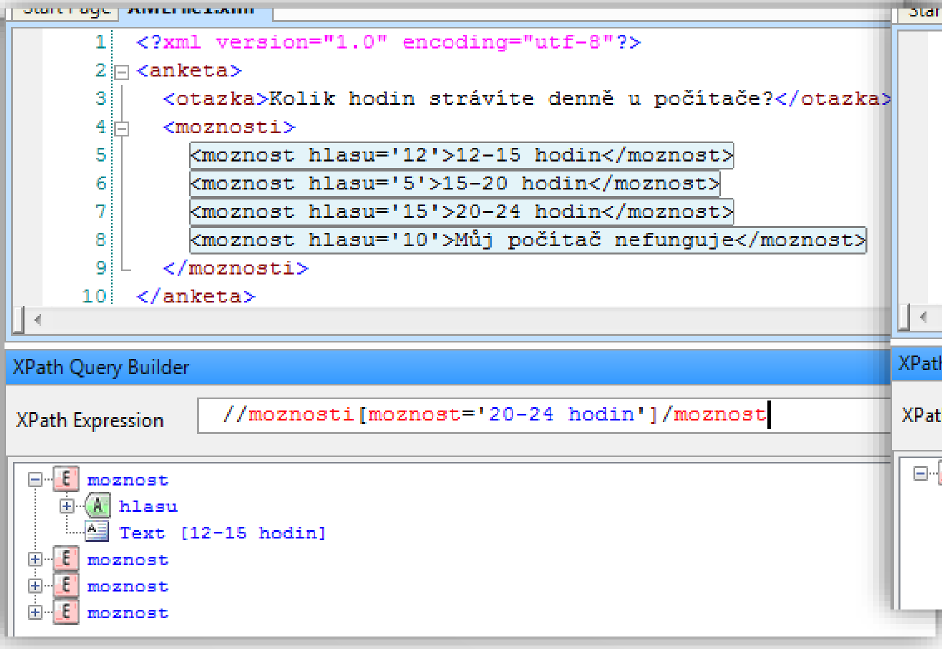
```
moznost
  hlasu
  Text [Můj počítač nefunguje]
```



```
<?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 Expression `//*[string-length(text())>20]`

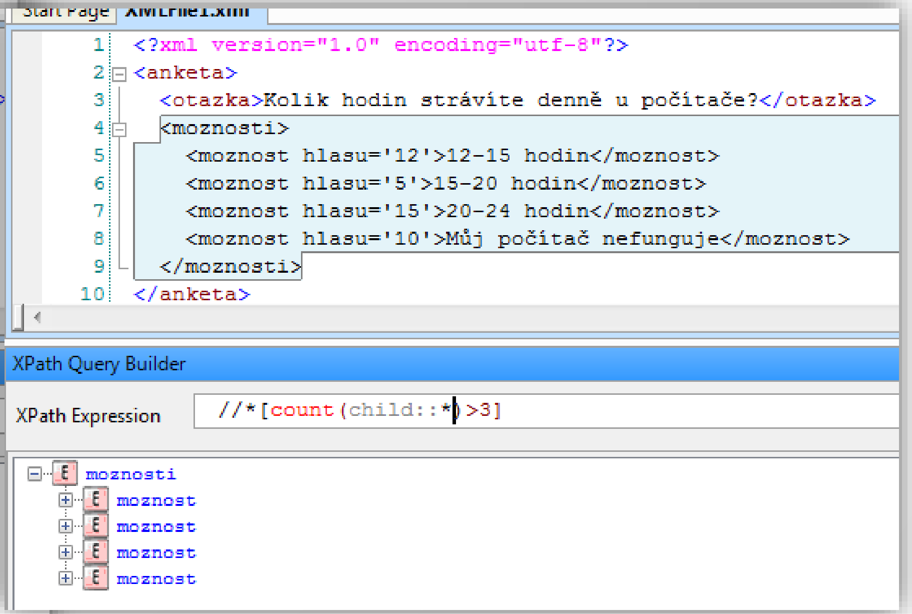
```
otazka
  Text [Kolik hodin strávíte denně u počítače?]
moznost
```



```
<?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 Expression `//moznosti[moznost='20-24 hodin']/moznost`

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

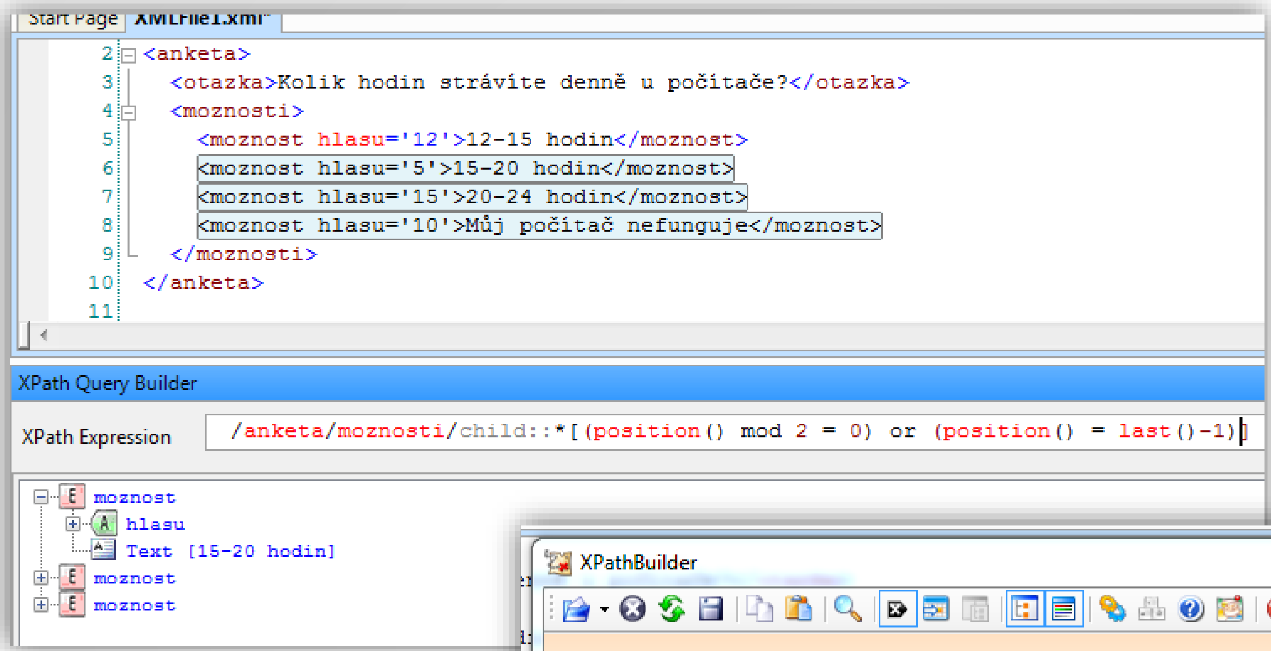


```
<?xml version="1.0" encoding="utf-8"?>
<anketa>
  <otazka>Kolik hodin strávíte denně u počítače?</otazka>
  <moznosti>
    <moznost hlasu='12'>12-15 hodin</moznost>
    <moznost hlasu='5'>15-20 hodin</moznost>
    <moznost hlasu='15'>20-24 hodin</moznost>
    <moznost hlasu='10'>Můj počítač nefunguje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

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

```
moznosti
  moznost
  moznost
  moznost
  moznost
```

XPath

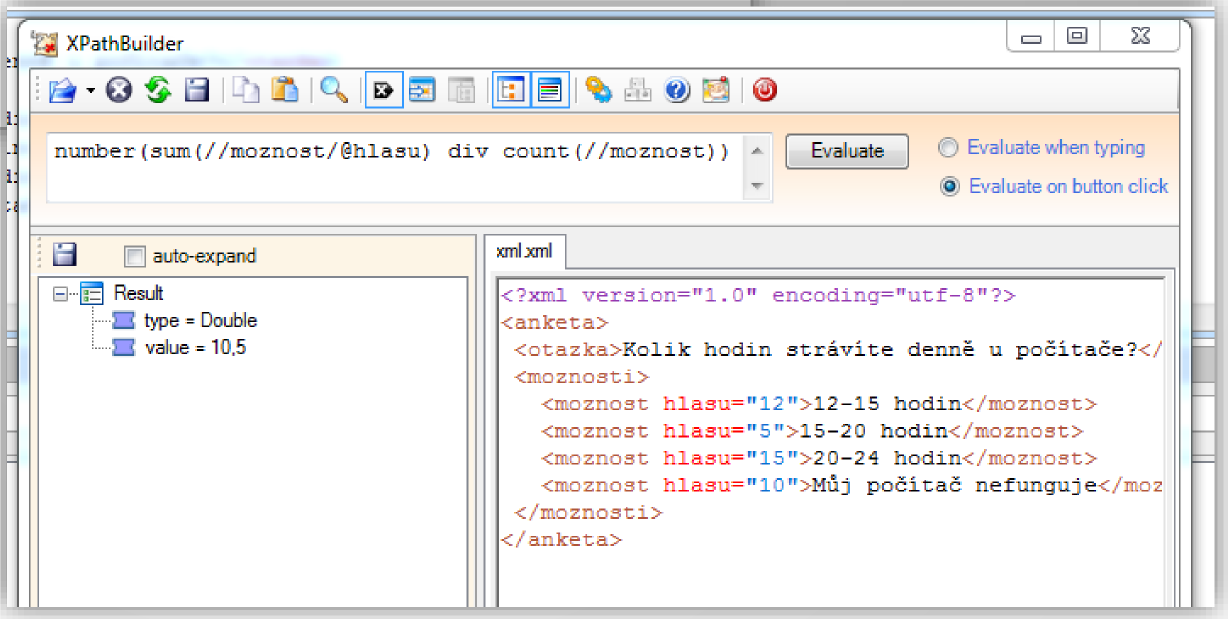


```xml
<anketa>
  <otazka>Kolik hodin strávite denně u počítače?</otazka>
  <moznosti>
    <moznost hlasu='12'>12-15 hodin</moznost>
    <moznost hlasu='5'>15-20 hodin</moznost>
    <moznost hlasu='15'>20-24 hodin</moznost>
    <moznost hlasu='10'>Můj počítač nefunguje</moznost>
  </moznosti>
</anketa>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

XPath Query Builder

XPath Expression: `/anketa/moznosti/child::*[(position() mod 2 = 0) or (position() = last()-1)]`

```
moznost
  hlasu
  Text [15-20 hodin]
moznost
moznost
```



`number(sum(//moznost/@hlasu) div count(//moznost))`

Evaluate when typing [✓ Evaluate on button click]

```
Result
  type = Double
  value = 10,5
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<anketa>
 <otazka>Kolik hodin strávite 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 and JavaScript

```
function loadXMLDoc(dname)
{
if (window.XMLHttpRequest)
  {
  xhttp=new XMLHttpRequest();
  }
else
  {
  xhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xhttp.open("GET",dname,false);
xhttp.send("");
return xhttp.responseXML;
}
xml=loadXMLDoc("books.xml");
path="/bookstore/book/title"
if (window.ActiveXObject)
{
var nodes=xml.selectNodes(path);

for (i=0;i<nodes.length;i++)
  {
  document.write(nodes[i].childNodes[0].nodeValue);
  document.write("<br>");
  }
}
// code for Mozilla, Firefox, Opera, etc.
else if (document.implementation && document.implementation.createDocument)
{
var nodes=xml.evaluate(path, xml, null, XPathResult.ANY_TYPE, null);
var result=nodes.iterateNext();

while (result)
  {
  document.write(result.childNodes[0].nodeValue);
  document.write("<br>");
  result=nodes.iterateNext();
  }
```

JSON

- JavaScript Object Notation
  - Data collection of pairs key/value
  - A list of values
  - Data types – JSONString, JSONNumber, JSONBoolean, JSONNull, etc.
- Suitable for exchange and transport of structured data

- [{"name": „BigBangTheory", "tvname": "CT1"}, {"name": "Comeback", "tvname": "Nova"}, {"name": „Friends", "tvname": "Prima"}]

- http://jsonlint.com/, http://braincast.nl/samples/jsoneditor/

JSON and JavaScript

```
function loadJSON()
{
   var data_file = "http://www.tutorialspoint.com/json/data.json";
   var http_request = new XMLHttpRequest();
   try{
      // Opera 8.0+, Firefox, Chrome, Safari
      http_request = new XMLHttpRequest();
   }catch (e){
      // Internet Explorer Browsers
      try{
         http_request = new ActiveXObject("Msxml2.XMLHTTP");
      }catch (e) {
         try{
            http_request = new ActiveXObject("Microsoft.XMLHTTP");
         }catch (e){
            // Something went wrong
            alert("Your browser broke!");
            return false;
         }
      }
   }
   http_request.onreadystatechange  = function(){
      if (http_request.readyState == 4  )
      {
        // Javascript function JSON.parse to parse JS
        var jsonObj = JSON.parse(http_request.responseText); //eval function deprecated

        // 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();
}
```

```
$(document).ready(function(){
  $("button").click(function(){
    $.getJSON("demo_ajax_json.js",function(result){
      $.each(result, function(i, field){
        $("div").append(field + " ");
      });
    });
  });
});
```