VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Web Application Development

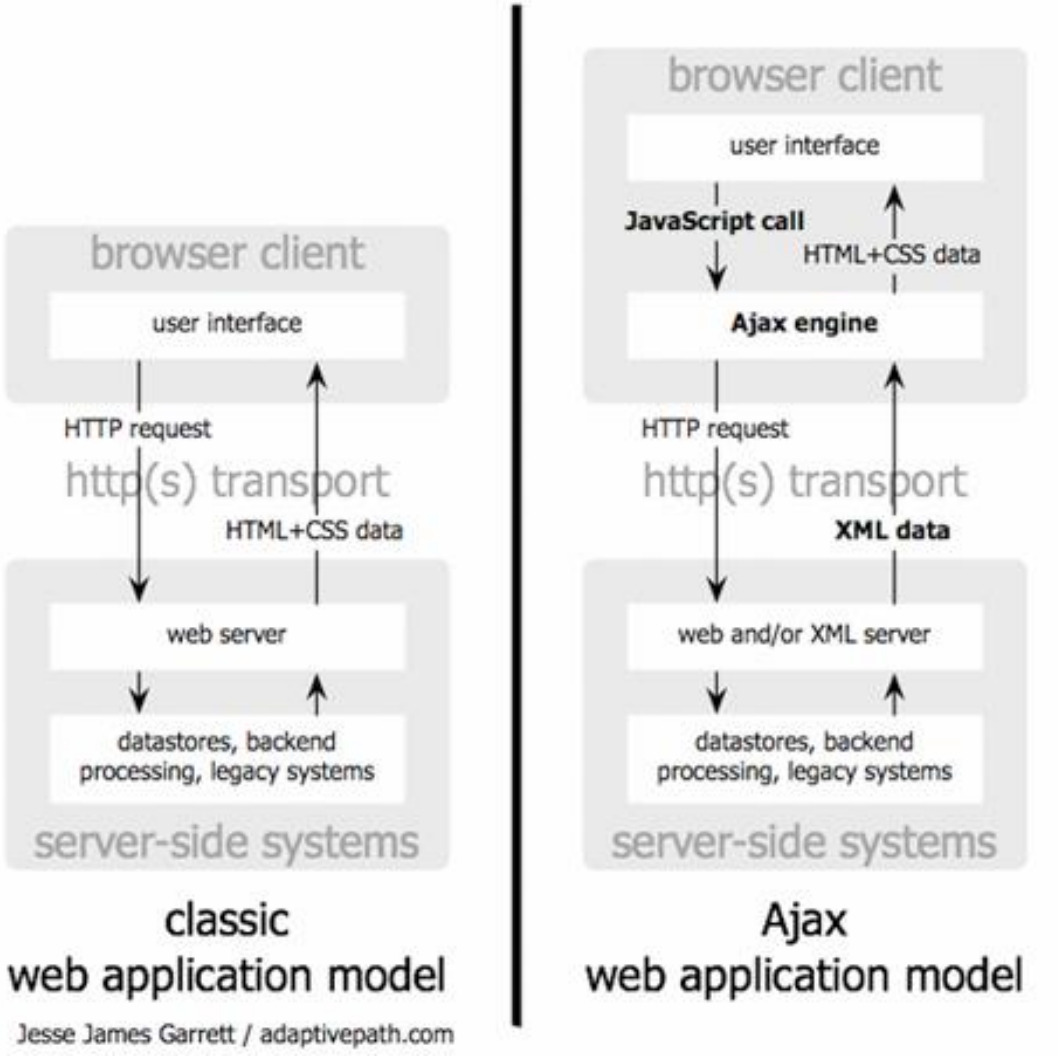
AJAX and XML

Ing. Michal Radecký, Ph.D.
www.cs.vsb.cz/radecky

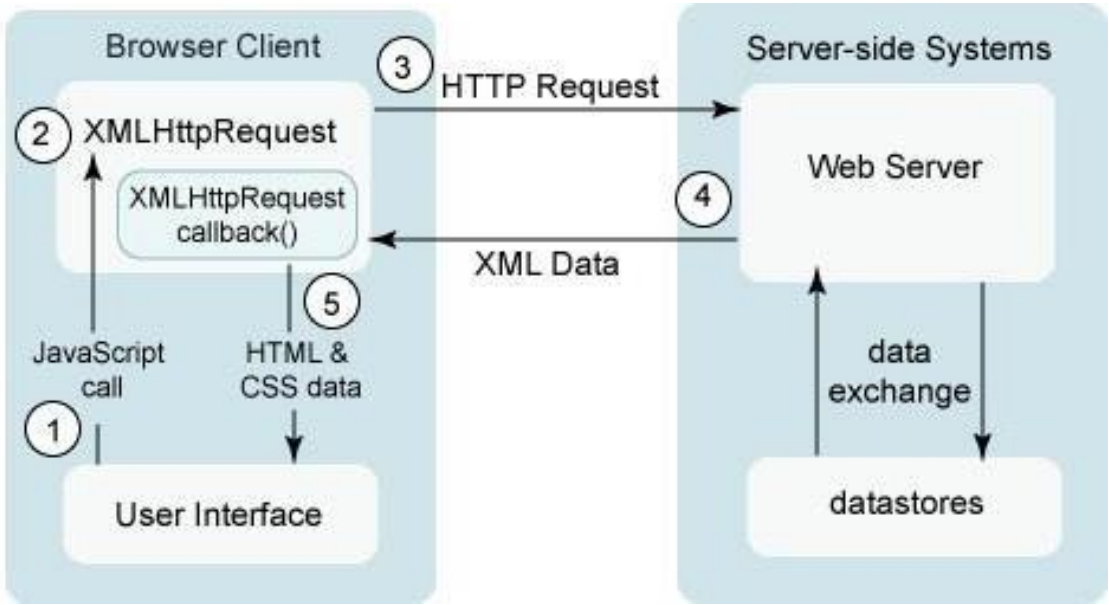
What is AJAX

- Asynchronous JavaScript and XML
- A combination of technologies that allows you to change parts of a web page depending on the data (invoking and processing HTTP requests), without the need to update the entire page.
- It is based on earlier ideas (IFRAME, LAYER, Applets, etc.), first described in the form used today in 2005
- Advantages
  - Greater user comfort and efficiency of using web applications
  - Lower demands on the amount of data transferred
- Disadvantages
  - Eliminate the functionality of the Back button in the browser
  - Changes inside the page do not affect the page itself (URL)

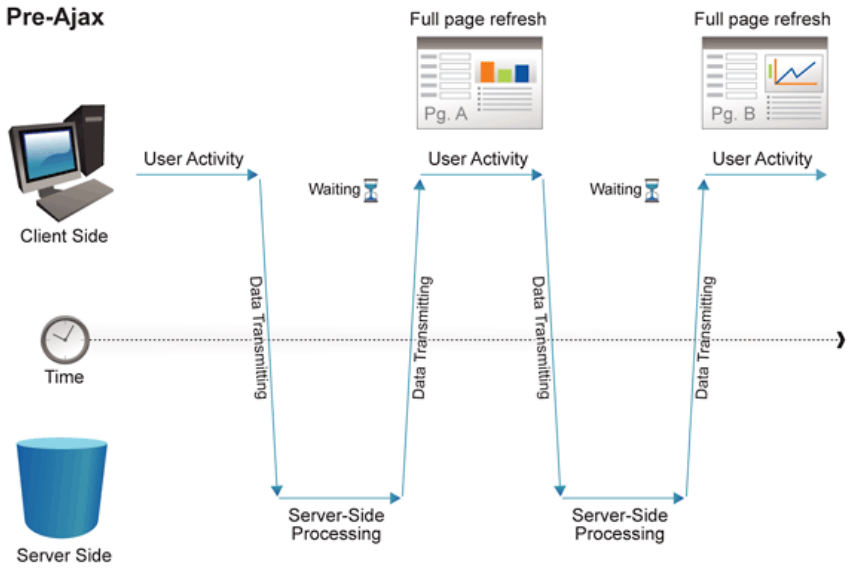
Operating model

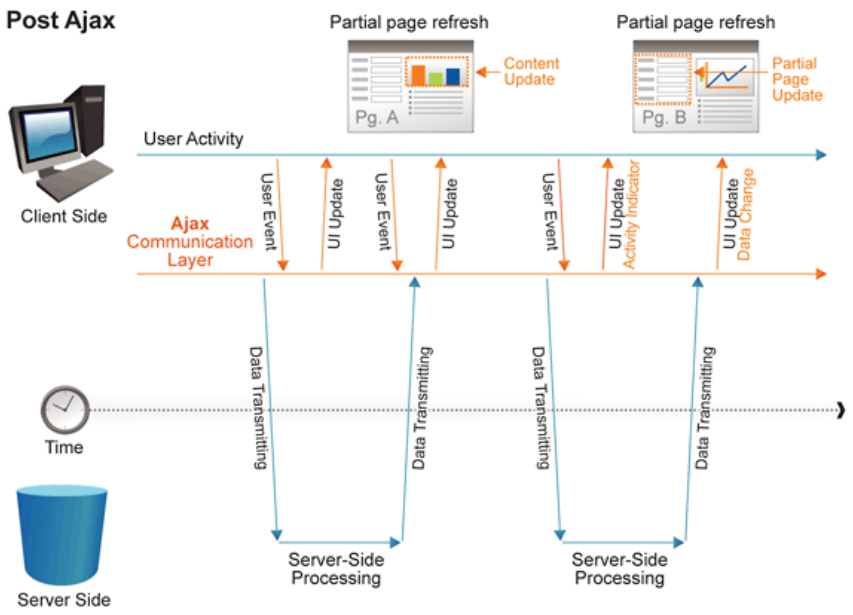


Zdroj: http://www.eioba.com



Operating model





Zdroj: http://www.websiteoptimization.com

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

AJAX and implementation

- DOM and XmlHttpRequest
- Possibility to use 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");
    }
 }
```

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

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

AJAX and implementation

- Fetch using asynchronous programming

```
fetch(url)
  .then(response => {
      if (!response.ok) {throw new Error('Network response was not ok');}
      return response.json();
      })

  .then(data => {
      console.log(data);
      })

  .catch(error => {
      console.error('There was a problem with the fetch operation:', error);
   });
```

Zdroj: http://www.ibm.com

AJAX and jQuery

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

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

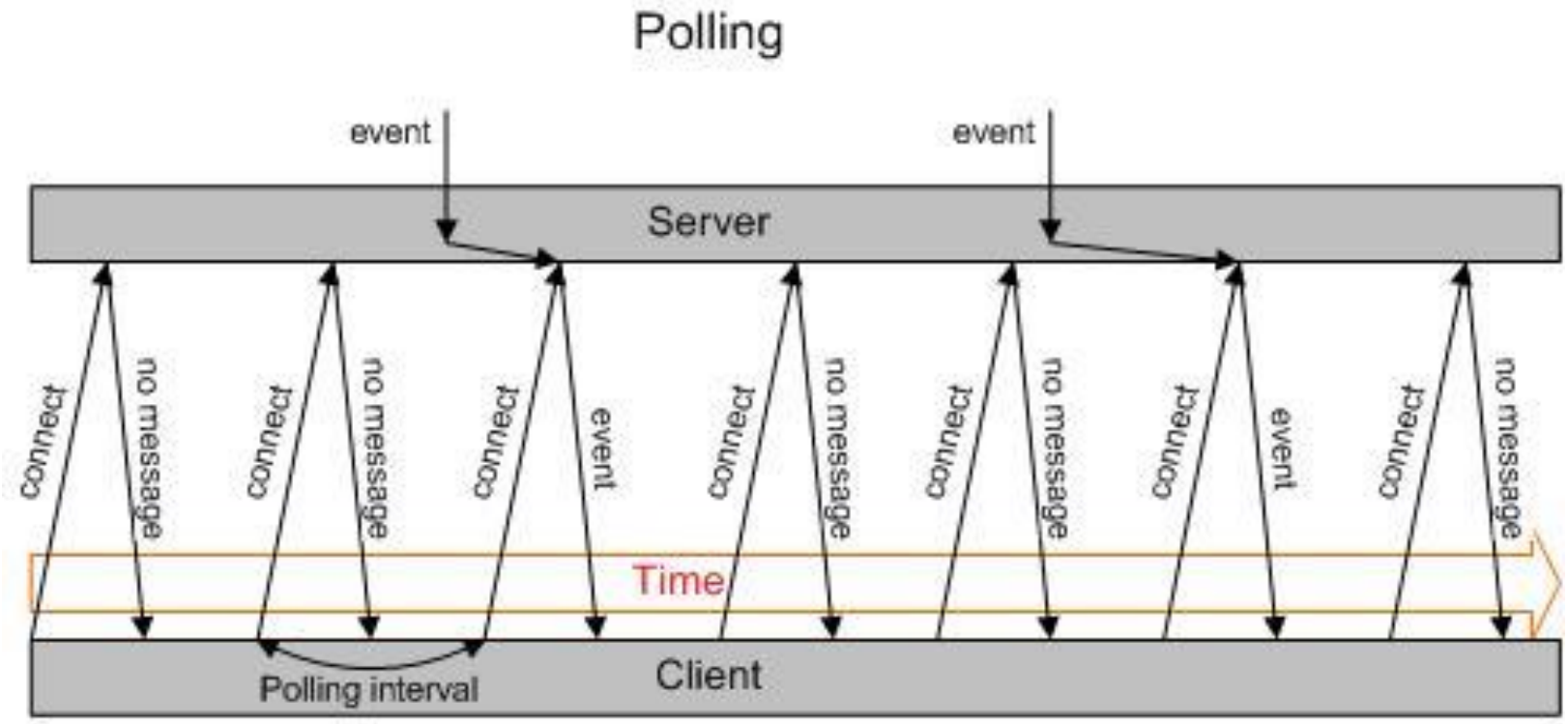
```
$.ajax({
    url: 'document.xml',
    type: 'GET',
    dataType: 'xml',
    timeout: 1000,
    error: function(){
        alert('Error loading XML document');
    },
    success: function(xml){
        $(xml).find('item').each(function(){
        var item_text = $(this).text();

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

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

Asynchronous approaches

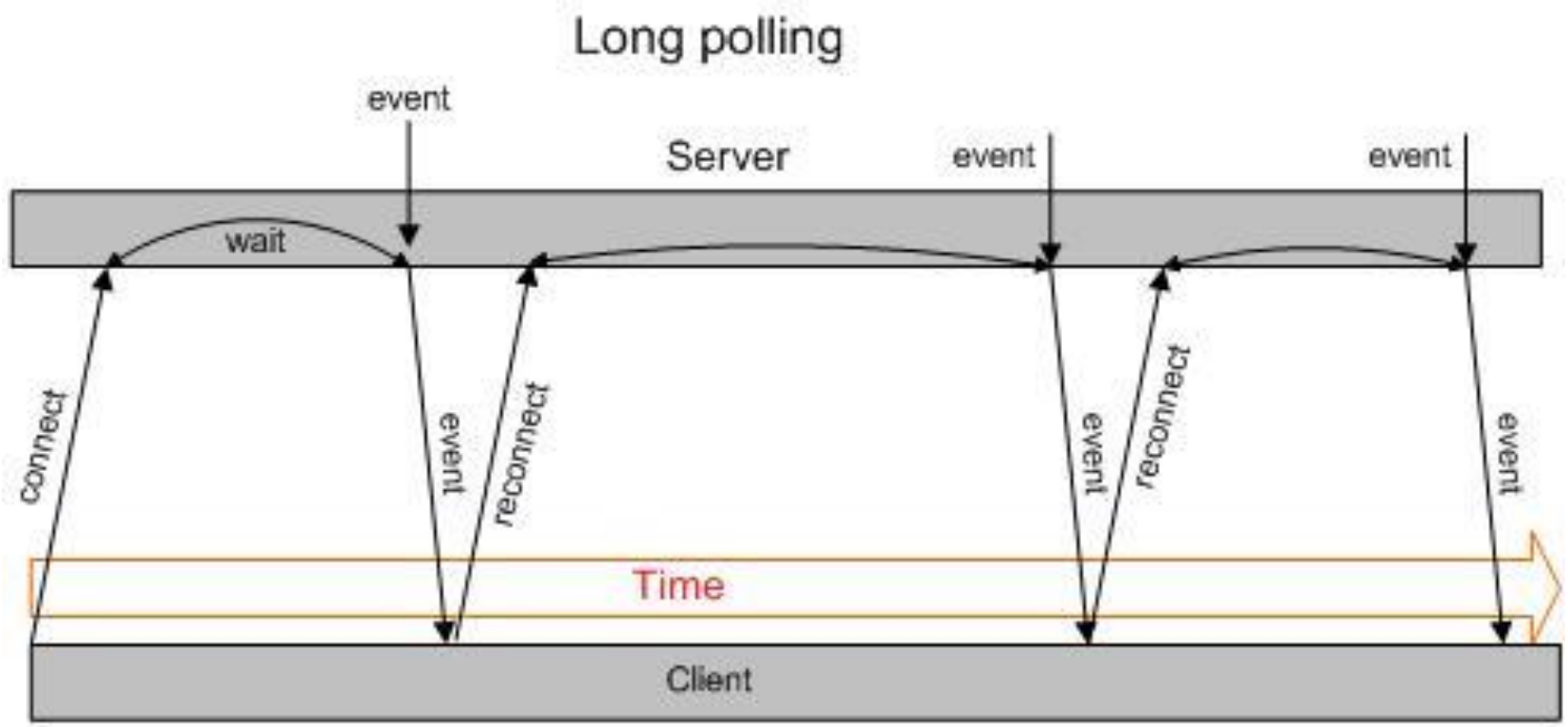
Polling



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

Asynchronous approaches

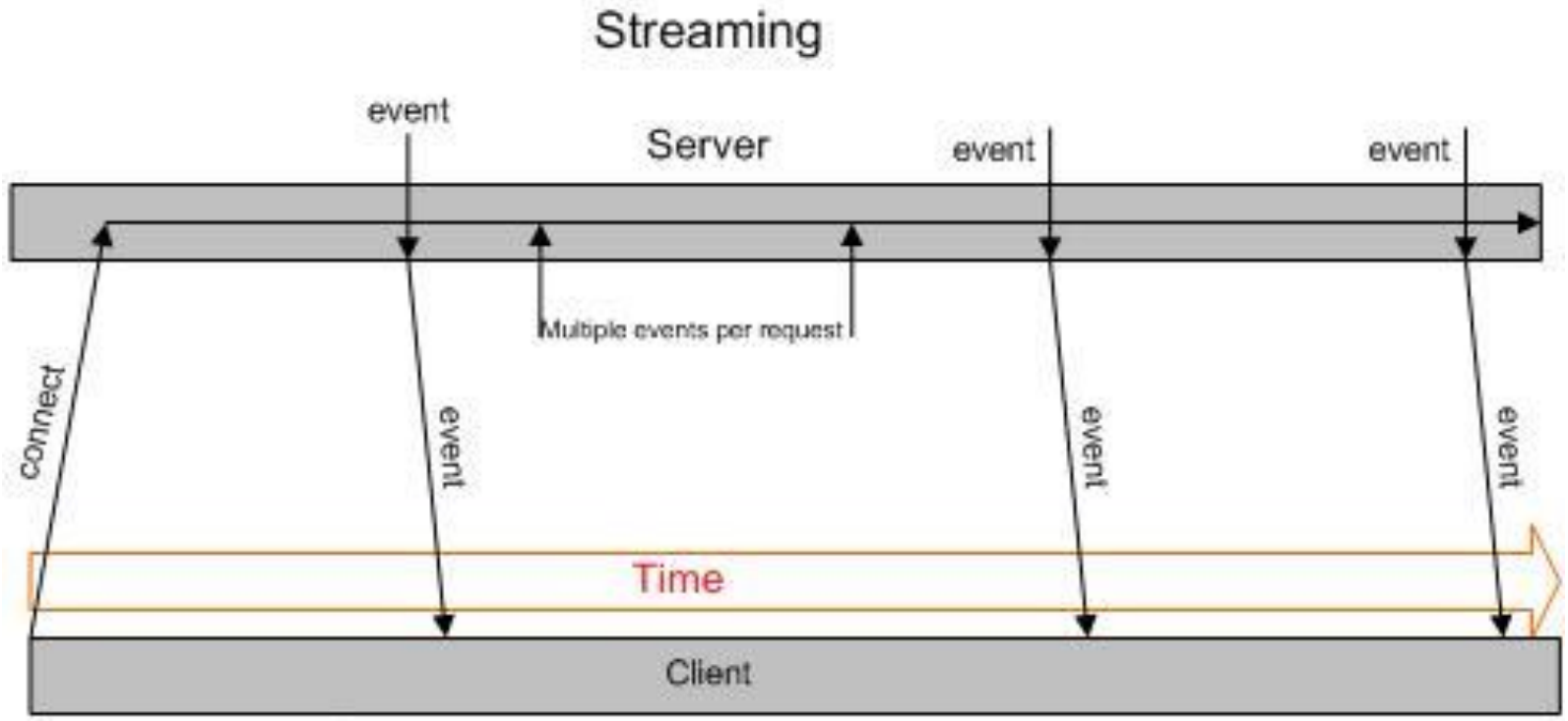
Long - polling



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

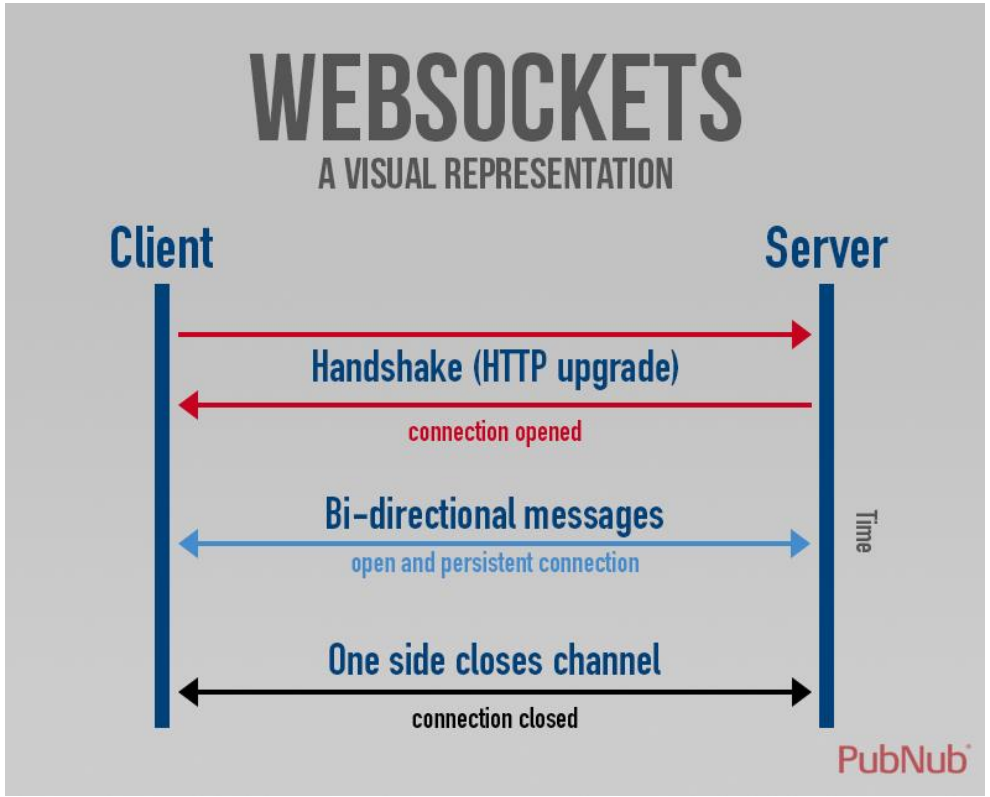
Asynchronous approaches

- Streaming
- Push approach
- Comet, reverse AJAX – Many implementations, different techniques



WebSockets

- Persistent communication bidirectional channel
- Specific WebSocket object
- send, onmessage, onopen, onerror, readyState



What is XML

- eXtensible Markup Language
- Ruleset
  - Semantic tags (tags, elements)
  - Divide a document into parts according to structure
  - Identification of parts of a document
- Language for descripting languages
  - Meta-branded language
  - defines the syntax for defining another language
- Based on SGML(Standard Generalized Markup Language)
  - Same options
  - Simplicity
- This is not another markup language
  - It's a meta-language
  - Specific names of elements, attributes, etc. are under the direction of the document creator

Why use XML

- Data + tags = structured data with semantics
- Allows you to define links (relations) between elements
- Can be 100% ASCII text
- Is documented in detail by the W3C
- Is not patented, has no copyright and other similar restrictions
- There are no versions of XML (as such)
- Support in programming languages
- Support in tools
- Simple processing

XML format

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

```
<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>
  ...
```

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

Other specifications

Comments

- “<!--” … “-->”

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

Text without interpretation

- section **CDATA**

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:name`**
- 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="">
      ...
```

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

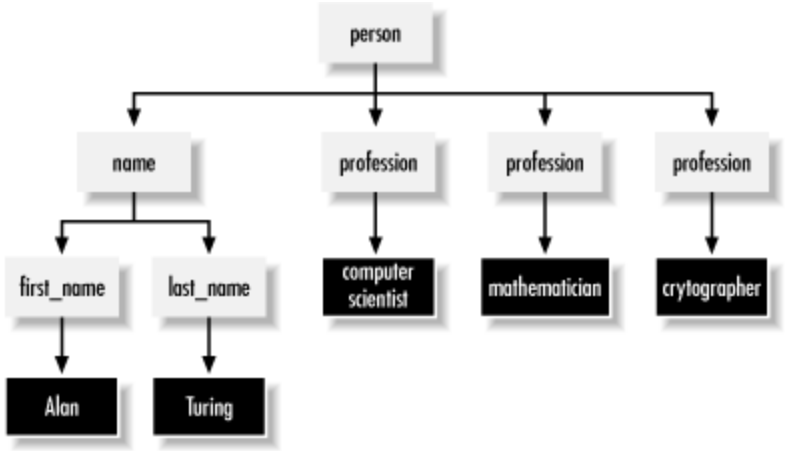
Parent, childs, …

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

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

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

Location

- In prolog after declaration
- Before first element

Directly DTD syntax or URL targeted DTD file

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

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

Usage of brackets for complex specifications

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>
```

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

DTD and XML

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)

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)

Zdroj: http://www.kosek.cz

XSD

```
<?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

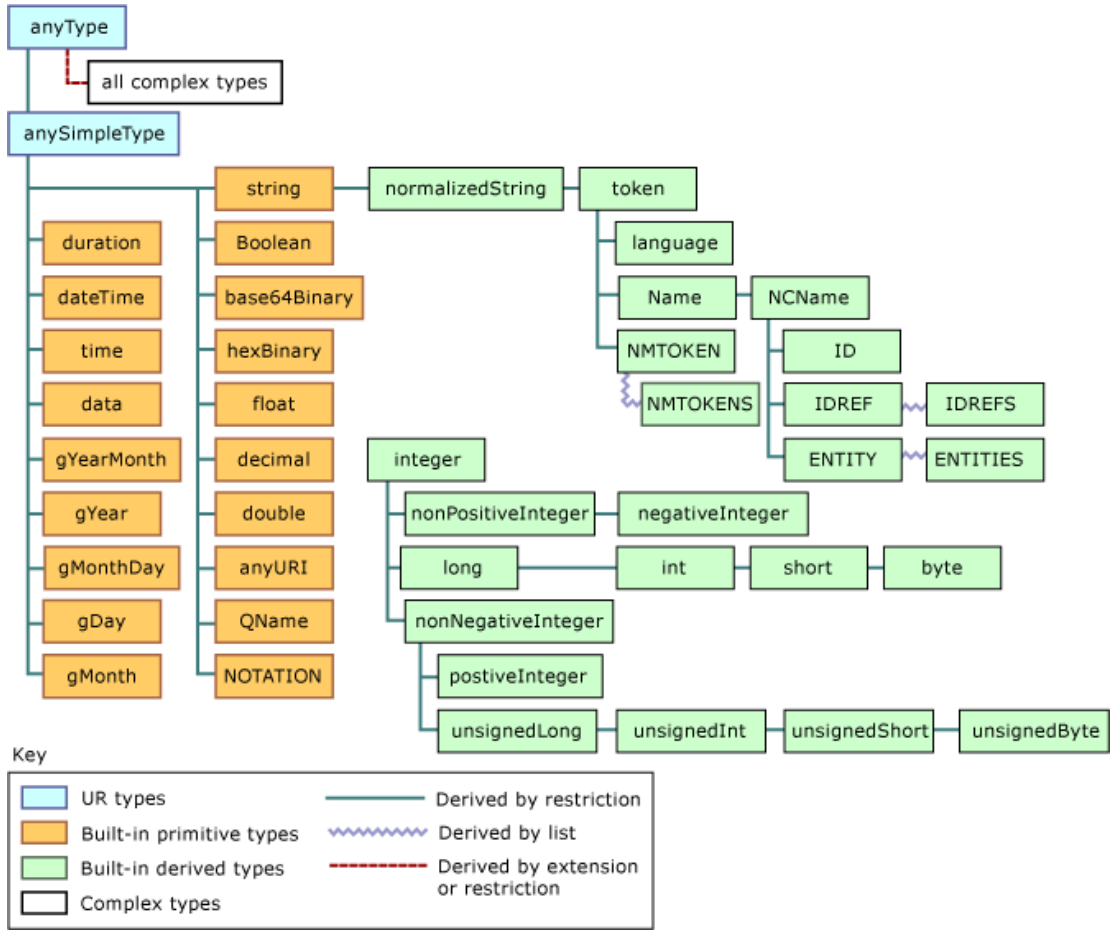
```
<xs:element name=„name" type=„type"/>
```

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

```
<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

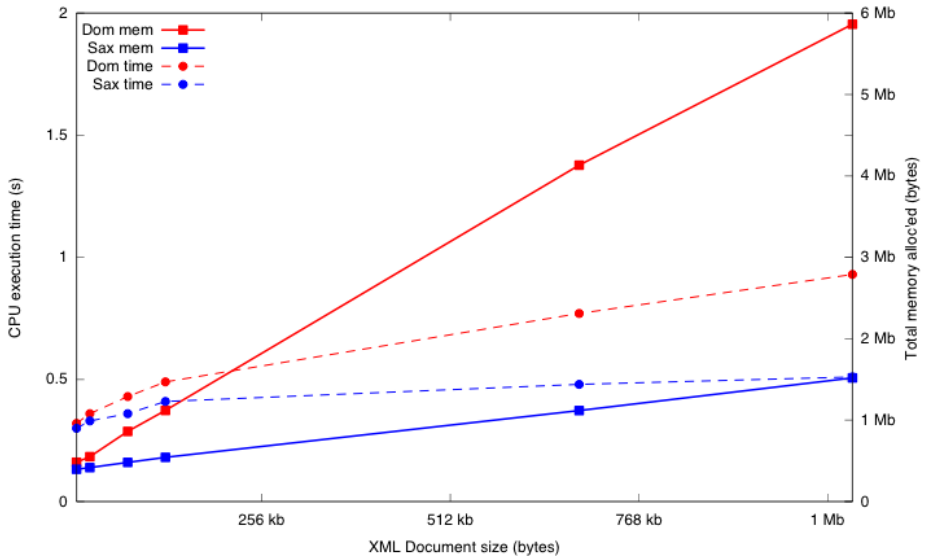
- Simple API for XML – event-driven model
- Processing of XML during its reading
- Mathod calling – processing data at the beginning/ending of some element, text content, etc.
- Fast, higher demands on implementation

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

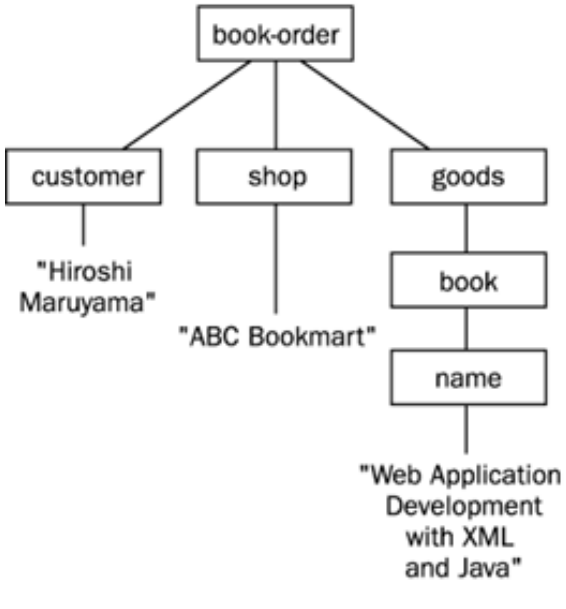
Zdroj: http://tech.inhelsinki.nl/2007-08-29/, http://book.javanb.com

DOM vs. SAX




```
<?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



DOM

```
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
```

SAX

JavaScript

From XML to DOM

```
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

Work with XMLDocument

- Same approach as DOM of whole web page
- Base is XMLDocument (Document)
- querySelector, querySelectorAll, getElement…
- createTextNode, createElement, appendChild, …

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

- It is possible to validate against DTD nor XSD
  https://vegibit.com/how-to-parse-and-generate-xml-in-javascript/