VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Vývoj Internetových Aplikací

AJAX a XML

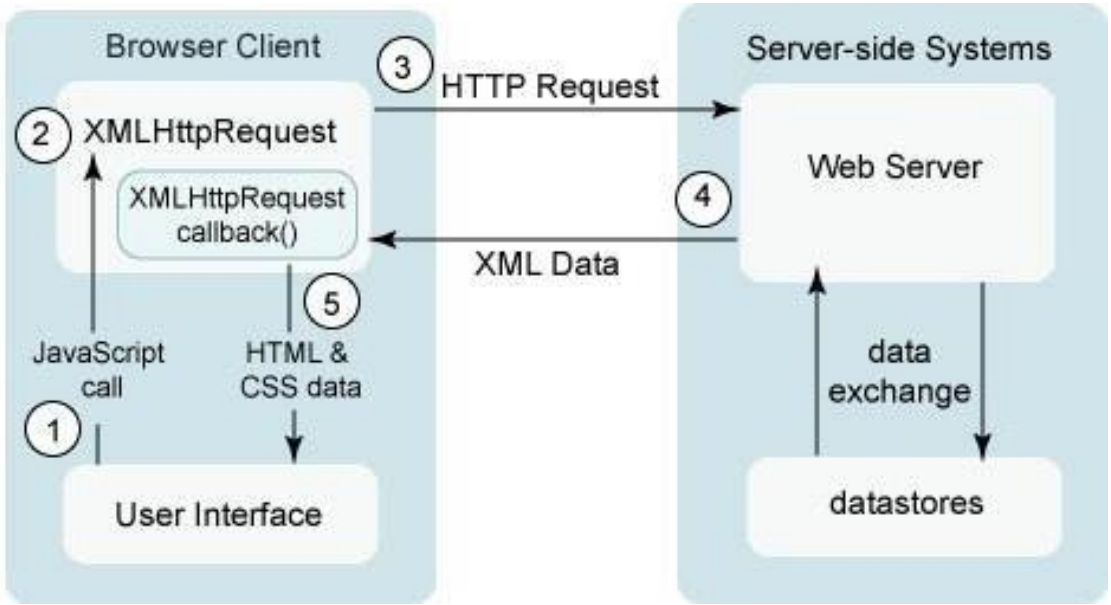
Ing. Michal Radecký, Ph.D.
www.cs.vsb.cz/radecky

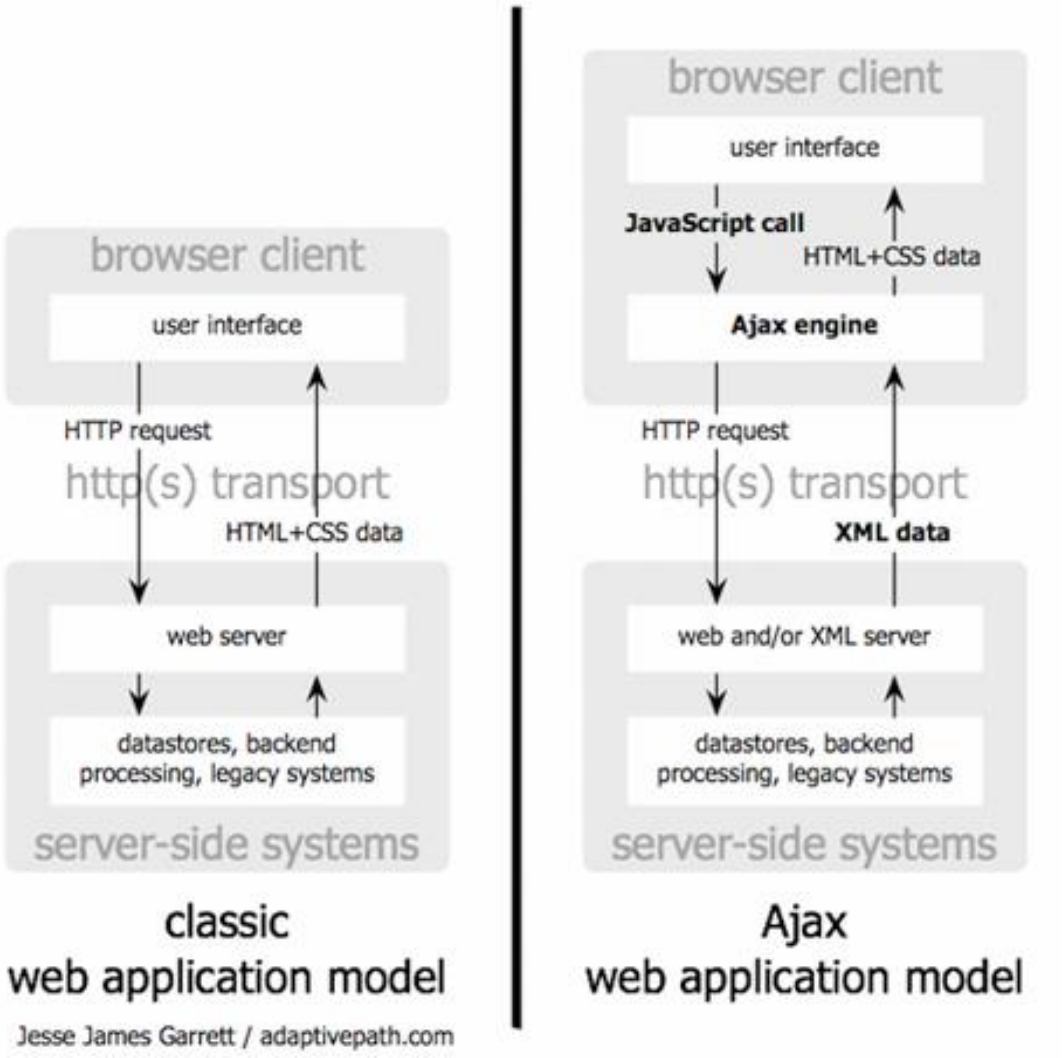
Co je to AJAX

- 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)

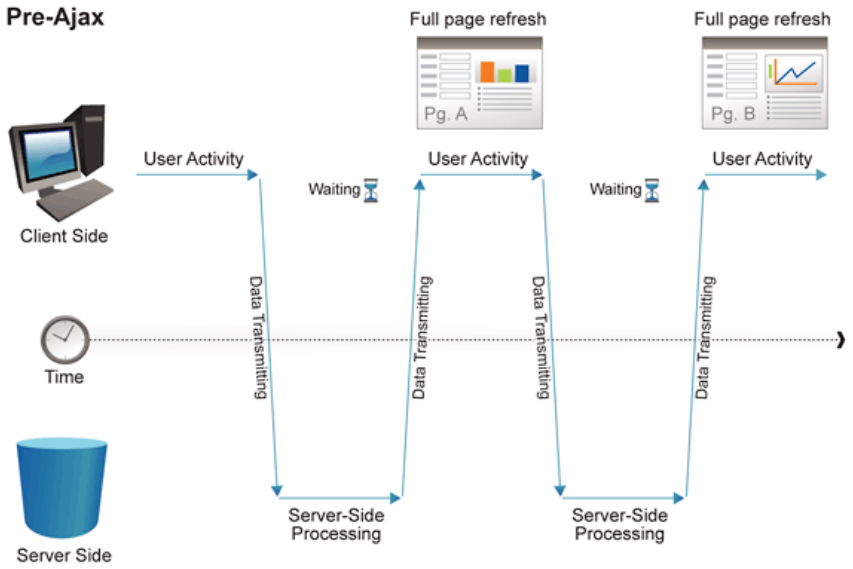
Model fungování

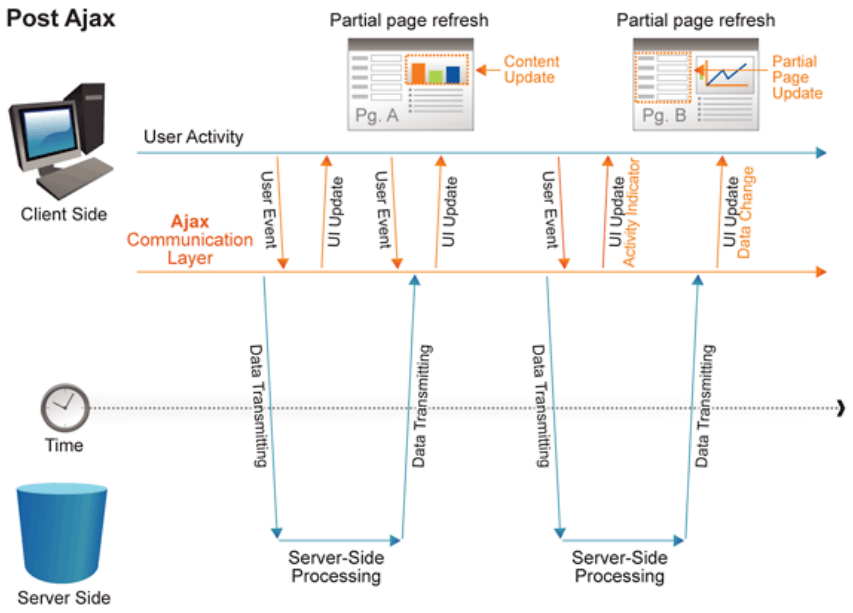
Zdroj: http://www.eioba.com





Model fungování





Zdroj: http://www.websiteoptimization.com

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

AJAX a implementace

- 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");
    }
 }
```

```
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 a implementace

- Fetch s využitím asynchronního programování

```
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 a jQuery

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

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

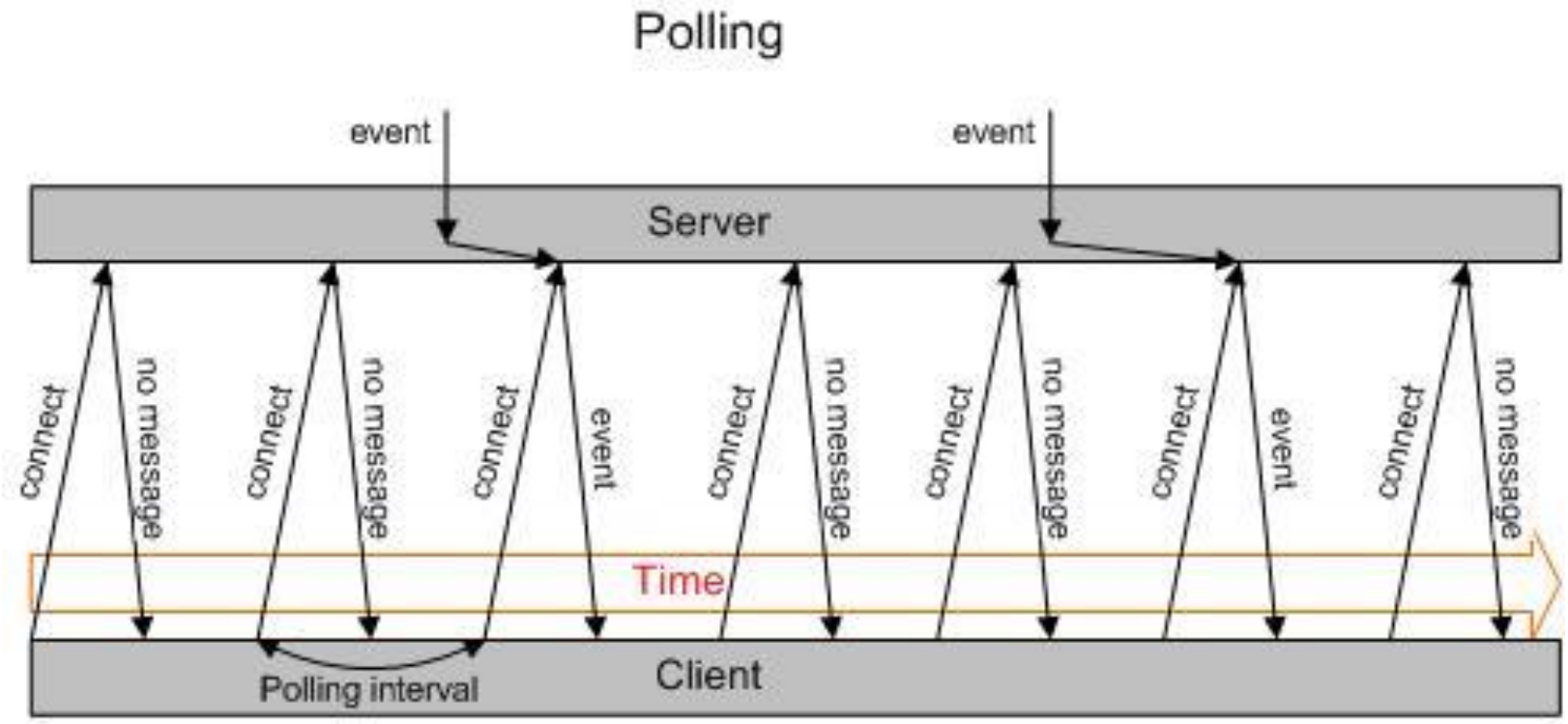
```
$.ajax({
    url: 'document.xml',
    type: 'GET',
    dataType: 'xml',
    timeout: 1000,
    error: function(){
        alert('Error loading XML document');
    },
    success: function(xml){
        $(xml).find('item').each(function(){
        var item_text = $(this).text();

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

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

Asynchronní přístupy

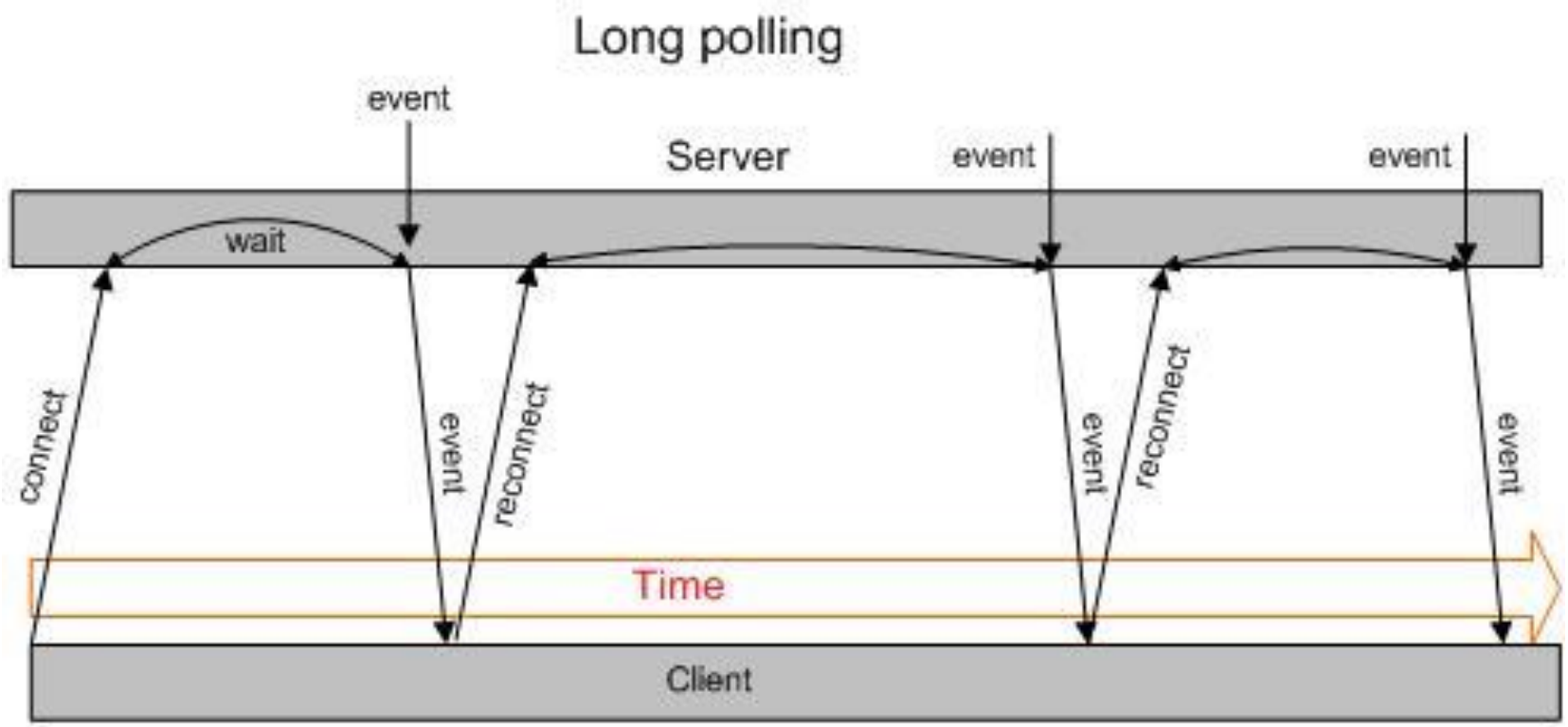
Polling



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

Asynchronní přístupy

Long - polling



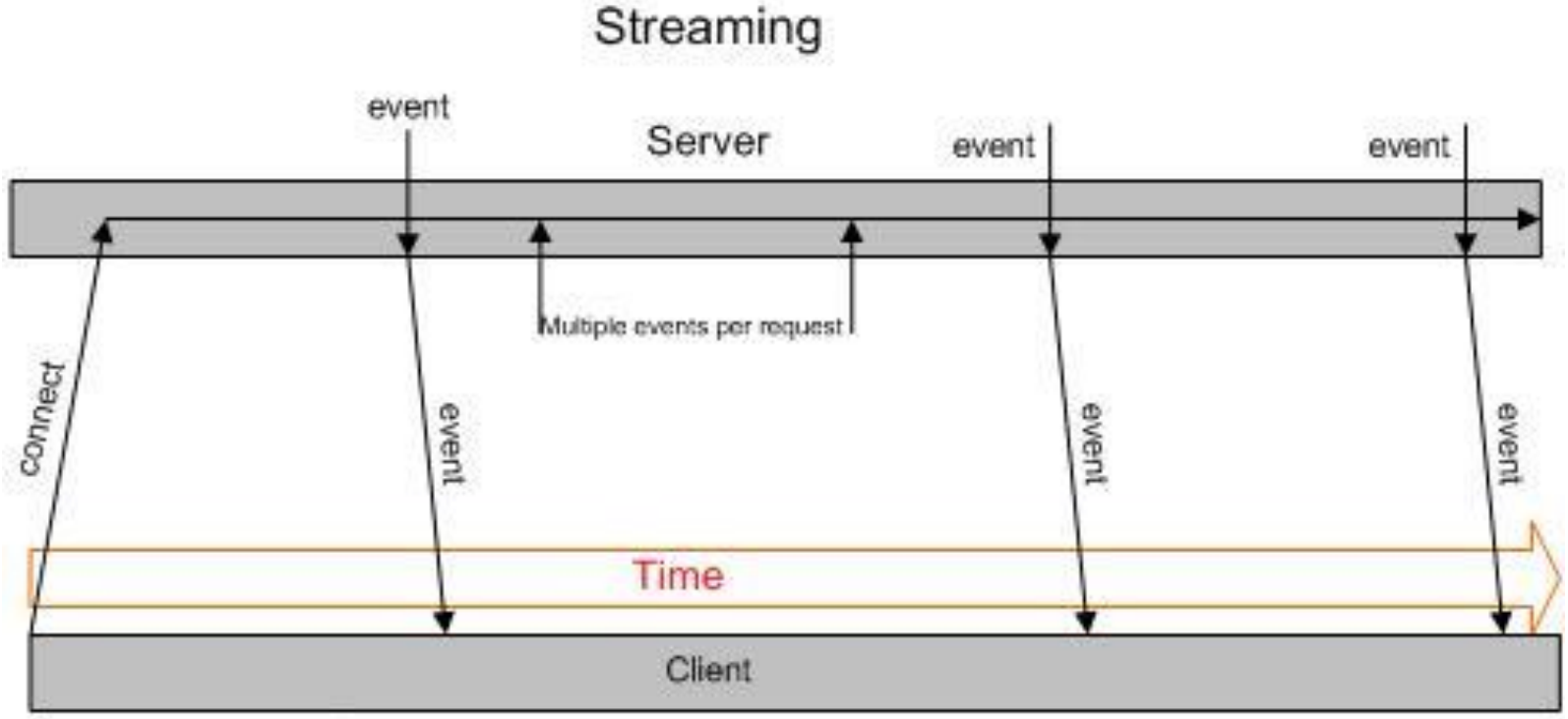
Zdroj: http://pic.dhe.ibm.com/infocenter

Asynchronní přístupy

Streaming

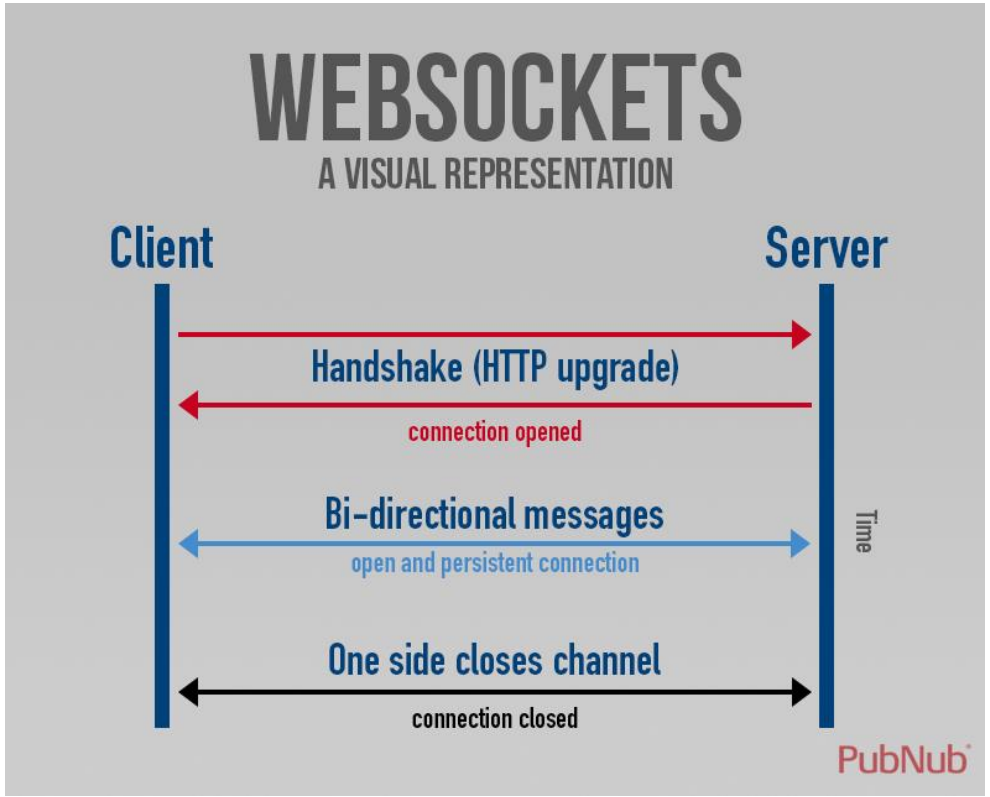
Push přístup

Comet, reverse AJAX – mnoho implementací. různé techniky



WebSockets

- Persistentní komunikační obousměrný 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

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

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

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

```
<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

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

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

Další specifikace

Komentáře

- začínají “<!--” a končí “-->”

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

Text bez interpretace

- sekce **CDATA**

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

- 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)

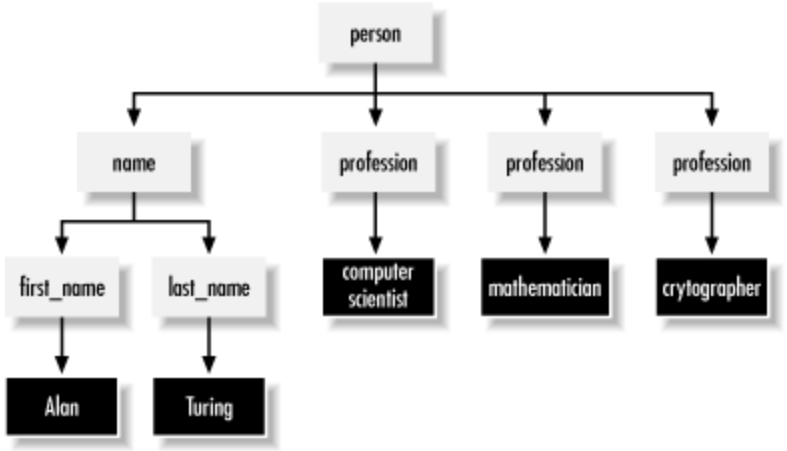
```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="keyword">
  ...
  </xsl:template>
</xsl:stylesheet>
```

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

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

Rodiče, děti, kořen, …

- 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>
```

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

DTD a XML příklad

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 for
   up-to-date data. The backup strategy includes running the database
   in Archive Log Mode and performing nightly backups. All new accounts
   need to be approved by the DBA Group before being created.
  </Comments>
 </DatabaseName>

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

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

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:

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

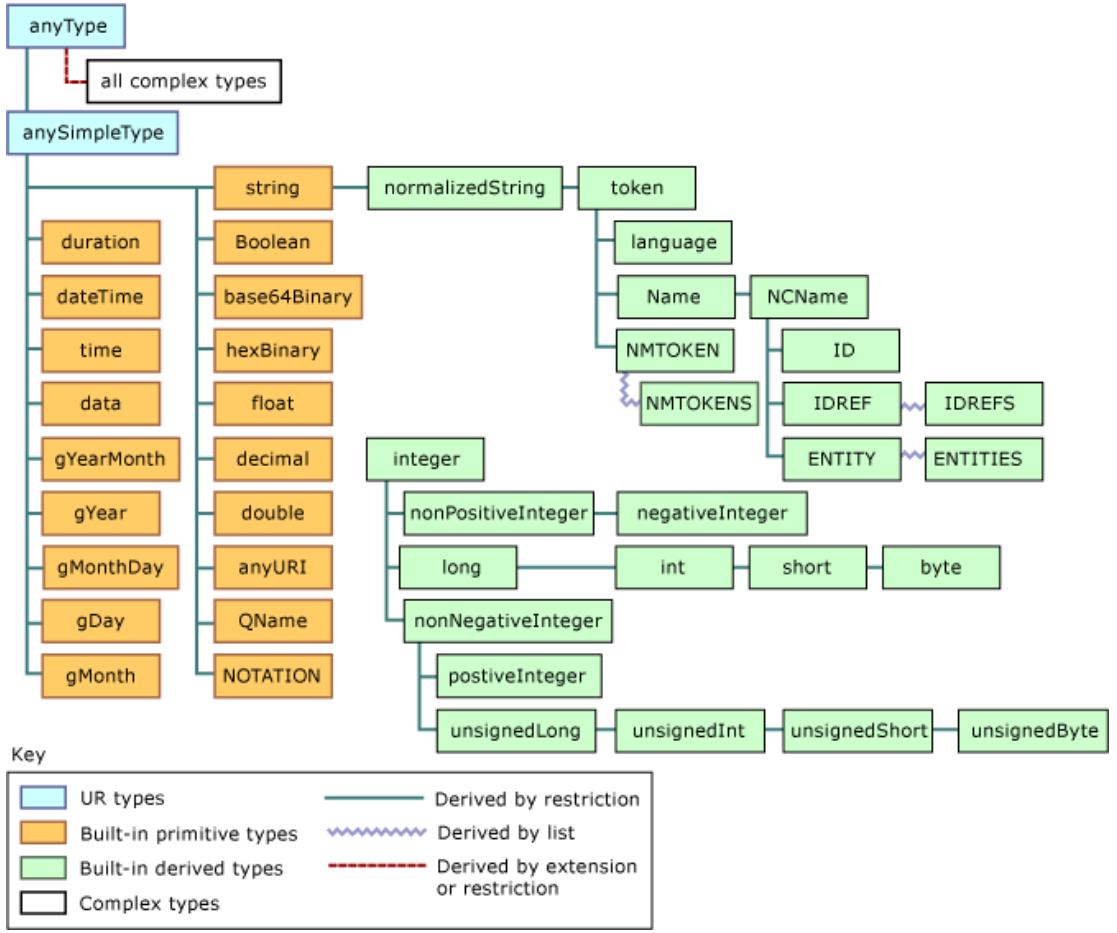
XSD – deklarace elementů

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

- Jméno odpovídá zvyklostem při vytváření jmen a proměnných
- Typ jeden z mnoha dostupných typů
- Možnost odvozování 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

```
<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

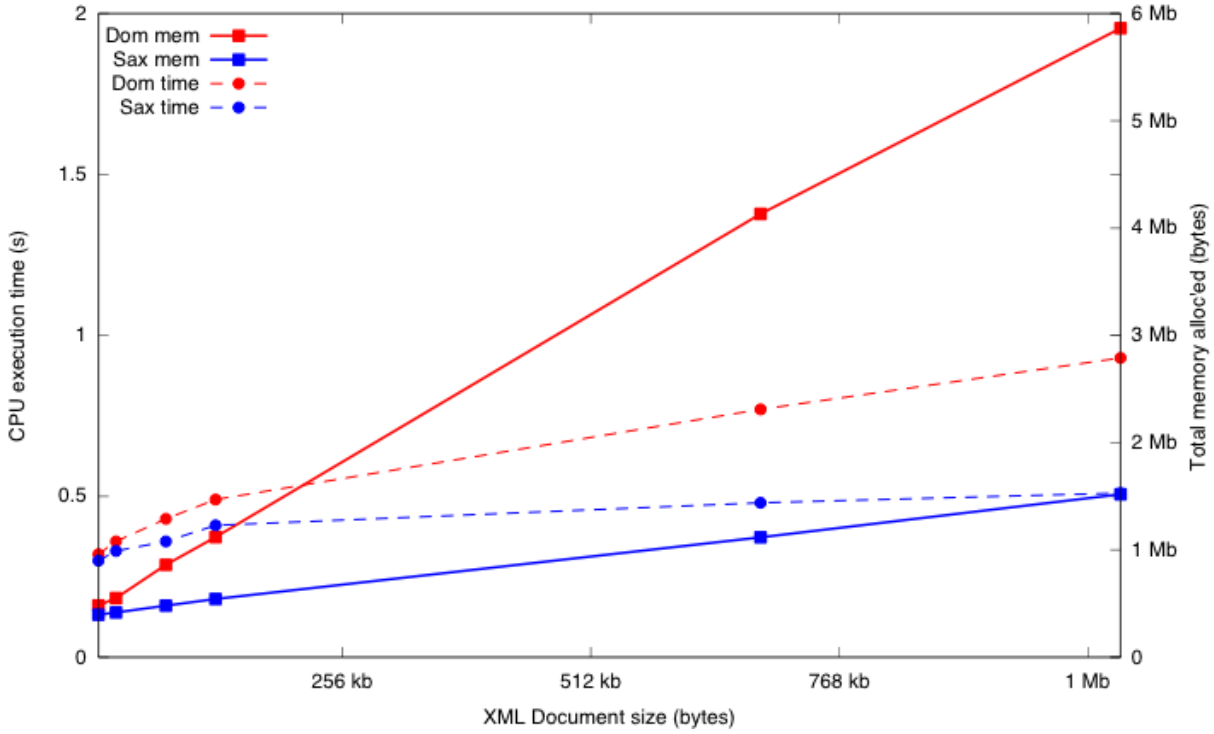
- Simple API for XML – event-driven model
- zpracování XML přímo při jeho načítání
- zpracování formou volání metod/zpracování načtených dat na začátku a konci elementů, při zpracování textu, apod.
- není přímo standardem
- rychlý, vyšší nároky na samotnou aplikaci

Parser

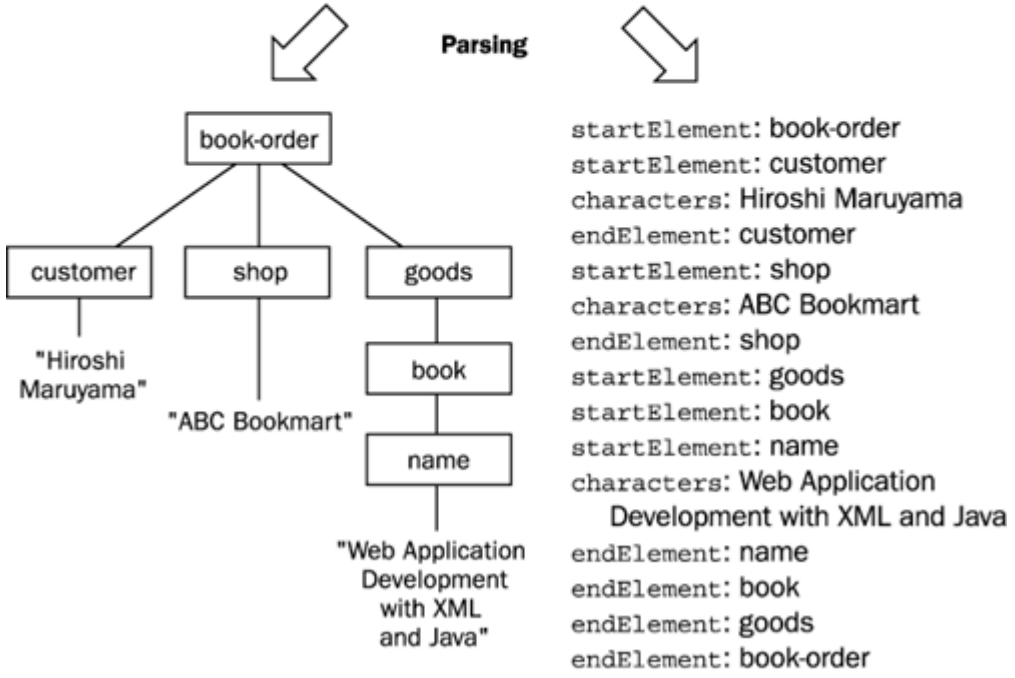
- aplikace, program, třída, algoritmus
- jeho úkolem je zpracovat XML dokument v textové podobě a převést jej do podoby dále zpracovatelné (např. DOM)
- ověřuje syntaxi, zajišťuje ověření DTD a validaci

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>
```



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/