

# Informační systémy

## AJAX

Martin Trnečka

Katedra informatiky  
Univerzita Palackého v Olomouci

# Předchozí přednášky

- Jazyk JavaScript nám umožňuje manipulovat s webovou stránkou na straně klienta.
- Jazyk PHP umožňuje webovou stránku generovat (celou nebo jen její část).
- Změna obsahu stránky = reload celé stránky.
- Změna obsahu, je nyní myšleno ve smyslu, kdy chceme načíst obsah ze serveru (obsah který není aktuálně k dispozici na straně klienta).
- Načíst celou stránku je občas nevhodné, například pokud chceme změnit jen jednu malou část webové stránky.
- Již dříve jsme mluvili o asynchronním načítáním zdrojů.
- Internetová revoluce: technologie AJAX.

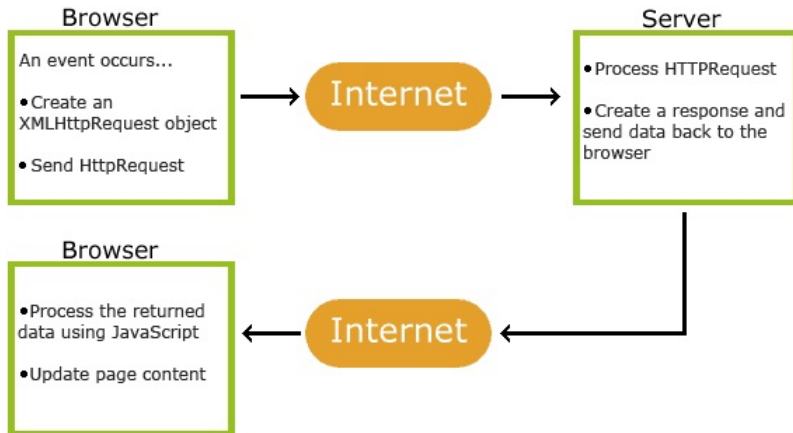
# AJAX

- Asynchronous JavaScript and XML.
- Striktně vzato to není nic nového. Nejedná se o nový jazyk, ale o způsob jak používat existující standardy.
- Určen pro výměnu dat ze serverem a změnu části stránky bez nutnosti reloadu celé stránky.
- Masivní využití AJAXu, například: Google Maps, Gmail, Youtube, Facebook.
- Poprvé představen Microsoftem v roce 2005 (prapůvod 1995-1998).
- V současnosti W3C připravuje XMLHttpRequest Level 1.

## Použité standardy:

- XMLHttpRequest objekt (asynchronní komunikace se serverem)
- JavaScript/DOM
- CSS
- XML (datový formát pro výměnu dat)

# Jak to funguje



# AJAX použití v HTML

## Příklad (Vytvoření XMLHttpRequest objektu)

```
1 var xmlhttp;
2 if (window.XMLHttpRequest) {
3     // code for IE7+, Firefox, Chrome, Opera, Safari
4     xmlhttp=new XMLHttpRequest();
5 }
6 else {
7     // code for IE6, IE5
8     xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
9 }
```

XMLHttpRequest představuje komunikační soket. Umožňuje odesílat a přijímat požadavky. Lze i komplikovaněji více na:

[https://developer.mozilla.org/en-US/docs/AJAX/Getting\\_Started](https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started)

# AJAX odeslání požadavku

## Příklad (Odeslání požadavku)

```
1 xmlhttp.open("GET", "url", true);  
2 xmlhttp.send();
```

Poslední parametr u `xmlhttp.open` udává zda má být komunikace asynchronní (`true`) nebo synchronní (`false`).

## Příklad (Odeslání požadavku)

```
1 xmlhttp.open("POST","demo_post.asp",true);  
2 xmlhttp.send();  
3  
4 // vyuziti nepovinného atributu funkce send (jen u POST požadavku)  
5 xmlhttp.open("POST","ajax_test.asp",true);  
6 xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");  
7 xmlhttp.send("fname=Henry&lname=Ford");
```

# AJAX přijetí požadavku

## Dva způsoby přijetí požadavku

1. Jako text.
2. Jako XML (DOM).

### Příklad (Přijetí požadavku)

```
1 document.getElementById("myDiv").innerHTML+xmlhttp.responseText;
2
3 xmlDoc+xmlhttp.responseXML;
4 txt="";
5 x=xmlDoc.getElementsByTagName("ARTIST");
6 for (i=0;i<x.length;i++)
7 {
8     txt=txt + x[i].childNodes[0].nodeValue + "<br>";
9 }
10 document.getElementById("myDiv").innerHTML=txt;
```

# Životní cyklus požadavku

Server odesílá průběžné informace o stavu požadavku (readyState).

## Stavy:

- 0: request not initialized
- 1: server connection established
- 2: request received
- 3: processing request
- 4: request finished and response is ready

Při změně stavu, spuštění triggeru onreadystatechange. Dále je uložena informace o statusu požadavku v proměnné status (200 - OK, 404 - soubor nenalezen).

## Příklad

```
1 xmlhttp.onreadystatechange=function() {  
2   if (xmlhttp.readyState==4 && xmlhttp.status==200) {  
3     document.getElementById("myDiv").innerHTML=xmlhttp.responseText;  
4   }  
5 }
```



# Vícenásobný http požadavek

- Objekt `xmlhttp` lze použít opakovaně k různým účelům.
- Nicméně pokud jej použijeme pro dva či více http požadavků, dříve dokončený http požadavek přepíše ostatní požadavky!
- Nedochozí ke ztrátě jako takové (první dokončený požadavek nastaví `readyState` na hodnotu 4). Na serveru se všechny požadavky vykonávají korektně.
- Řešení: zjednodušeně můžeme vidět `xmlhttp` objekt jako obsluhu pro jeden požadavek. Pokud chceme více požadavků stačí vytvořit více `xmlhttp` objektů.

# AJAX v jQuery

- Velice rozsáhlá podpora.
- Celá řada základních (\$.get, \$.post, \$.ajax) i pokročilých funkcí (registrace callback funkcí). Podpora formátu JSON.

## Příklad

```
1 jQuery.ajax({
2     url: "skript.php",
3     type: "POST",
4     data: "data1=value1&data2=value2",
5     cache: false ,
6     dataType: "xml",
7     success: function(result) {
8         alert ( result );
9     }
10 });
```

Zjednodušená dokumentace:

[http://www.w3schools.com/jquery/ajax\\_ajax.asp](http://www.w3schools.com/jquery/ajax_ajax.asp)

# Formát JSON

- JavaScript Object Notation
- Zápis dat nezávislý na platformě (analogie XML).
- Minimalistický (protiklad XML), původně navržen pro real-time systémy.
- PHP funkce `json_decode()`, `json_encode()`, `json_last_error()`.

## Příklad

```
1 {  
2   "firstName": "Bart",  
3   "lastName": "Simpson",  
4   "address": {  
5     "streetAddress": "742 Evergreen Terrace",  
6     "city": "Springfield"  
7   },  
8   "phoneNumbers": [  
9     { "type": "home", "number": "555-1234" },  
10    { "type": "fax", "number": "555-4567" }  
11  ]  
12 }
```

# Kompletní příklad na AJAX

## Příklad (HTML stránka)

```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5   <title>jQuery Ajax Example with JSON Response</title>
6   <script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"
7     type="text/javascript"></script>
8 </head>
9
10 <body>
11   <a href="#" class="loadData" data-info="3">Load all data</a>
12   <a href="#" class="loadData" data-info="2">Load Marvel data</a>
13   <a href="#" class="loadData" data-info="1">Load DC data</a>
14
15   <div id="data">
16     <!-- Javascript will print data here -->
17   </div>
18
19 </body>
20 </html>
```

## Příklad (jQuery kód I.)

```
1 $(document).ready(function(){
2
3   $('loadData').click(function() {
4     var data_info = $(this).data("info");
5     $.ajax({ // ajax call starts
6       url: 'serverside.php', // JQuery loads serverside.php
7       data: 'data=' + data_info, // Send value of the clicked button
8       dataType: 'json', // Choosing a JSON datatype
9       success: function(data) // Variable data contains the data we get from serverside
10      {
11
12        $('#data').html(''); // Clear #data div
13
14        if (data_info == '3') { // If clicked value is 3
15          for (var i in data.marvel) {
16            $('#data').append('Marvel: ' + data.marvel[i] + '<br/>');
17          }
18          for (var i in data.dc) {
19            $('#data').append('DC: ' + data.dc[i] + '<br/>');
20          }
21        }
22      }
23    }
24  })
25 }
```

## Příklad (jQuery kód II.)

```
21     else if (data_info == '2') { // If clicked value is 2
22         for (var i in data) {
23             $('#data').append('Marvel: ' + data[i] + '<br/>');
24         }
25     }
26
27     else if (data_info == '1') { // If clicked value is 1
28         for (var i in data) {
29             $('#data').append('DC: ' + data[i] + '<br/>');
30         }
31     }
32 }
33 });
34
35 return false;
36 });
37 });
```

## Příklad (PHP kód, serverside.php)

```
1 // Get value of data—info
2 $data = $_GET['data'];
3
4 // Red wine table
5 $marvel = array('Iron Man', 'Avengers', 'Thor');
6 $dc = array('Superman', 'Batman', 'Green Arrow');
7
8 // Combine data tables into one multidimensional table
9 $all = array(
10     "marvel" => $marvel,
11     "dc" => $dc,
12 );
13
14 if ($data == "1") {
15     print json_encode($dc);
16 }
17 else if ($data == "2") {
18     print json_encode($marvel);
19 }
20 else {
21     print json_encode($all );
22 }
```

# Kritika AJAX


- Zvýšení počtu http požadavků (sice malé, ale analogie s kopírováním dat).
- Narušení struktury zpět a vpřed (dnes není již tak dogmatické).
- Nelze plnohodnotně zrekonstruovat „naklikanou“ webovou stránku (lze, ale dá to práci).
- Funguje jen v moderních prohlížečích.




## Důležité pojmy:

- Technologie AJAX a její použití.

## Čtení na doma:

 Ullman L., PHP and MySQL for Dynamic Web Sites, 4th Edition, 2012, ISBN 978-0-321-78407-0. (kapitola 15).

 <http://www.w3schools.com/ajax/default.asp>