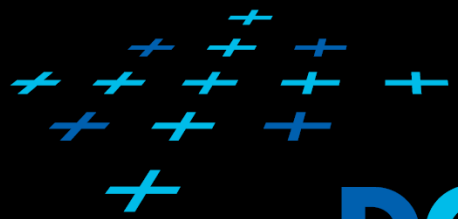


DCGI

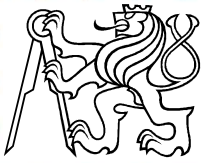
KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

Ajax v PHP

Martin Klíma

AJAX – co to je?

- **A**synchronous **J**avascript **A**nd **X**ML
- Webový klient komunikuje s webovým serverem asynchronně.
- Výsledkem je jen částečná aktualizace stránky
- Blíží se návrhu klasické desktopové aplikace
- AJAX není nová technologie
- Jde o novou aplikaci existující technologie, resp. o rádoby novinku v souvislosti s pojmem Web 2.0



Ajax

Příklady:

- Našeptávače (www.seznam.cz)
- On-line mapy (mapy.seznam.cz)
- gmail
- …

AJAX - jak to funguje

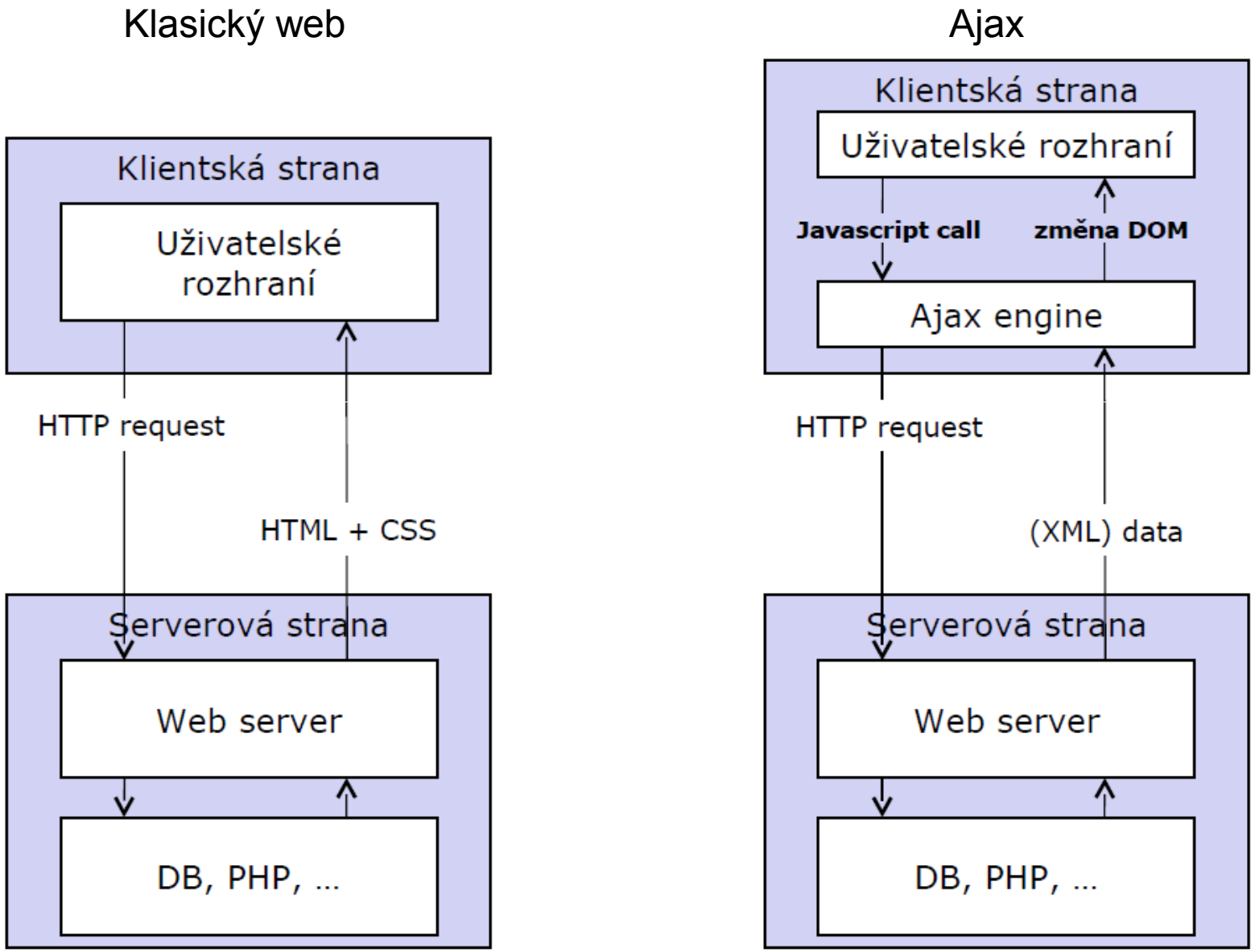
Klasický web

- Událost v prohlížeči vyvolá HTTP request
- Server obslouží dotaz a vrací nová data, např. HTML
- Klient dekóduje data a nahrazuje jimi celou stránku

Ajax

- Událost v prohlížeči je obsloužena javascriptovým handlerem
- Je vytvořen HTTP request a v něm předány informace
- Informace jsou zakódovány v dohodnutém formátu
- Server obslouží dotaz a vrací data
- Klient dekóduje data a modifikuje DOM

Ajax – jak to funguje II



Implementace

Několik různých způsobů implementace, všechny mají následující kroky

1. Otevři asynchronní spojení klient – server
2. Pošli dotaz pomocí domluveného protokolu
3. Zpracuj dotaz a manipuluj DOMem

Příklad implementace I

- Přidávání elementu, pro jehož získání je třeba vygenerovat http request

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <script language="JavaScript" type="text/javascript">
       function obrazky () {
        var obrazek= new Image(400,353);
        obrazek.src="obrazky/pejsek.jpg";
        document.getElementById("obr_id").appendChild(obrazek);
    }
    </script>
    <title>Obrázek</title>
  </head>
  <body>
  <form action="">
    <input type="button" value="Obrázek" onclick="obrazky();"/>
    <div id="obr_id"></div>
  </form>
  </body>
</html>
```

kuk ajax1_img.html

Příklad implementace II

- Přidávání obrázku je hezké, ale není v tom žádná logika ze serveru
- Nyní použijeme jiný element, který má také atribut *src* a který může zužitkovat serverovou logiku

Použití elementu SCRIP

1. Javascriptem vyrobíme nový element SCRIPT
2. Přiřadíme mu vlastnost src
   - to způsobí nahrání obsahu scriptu z externího zdroje
   - pokud je zdroj pod naší kontrolou, máme vyhráno
3. Skript přidáme do dokumentu

Příklad implementace II

kuk ajax2_script.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Ajax pomocí objektu SCRIPT</title>
     <script type="text/javascript">//<![CDATA[
          function vyrobDotaz() {
            var oScript = document.createElement("script");
            oScript.src = "skript_generovany_php.php";
            document.body.appendChild(oScript);
          }

          function vypisHodiny(hodiny_string) {
            document.getElementById("div_hodiny").innerHTML += "<br/>" +
hodiny_string;
          }
     //]]>
     </script>
   </head>
   <body>
    <form action="">
     <input type="button" value="Kolik je hodin?" onclick="vyrobDotaz()" />
     <div id="div_hodiny"></div>
    </form>
   </body>
</html>
```

skript_generovany_php.php

```
<?php
echo "vypisHodiny(\"".date("H:i:s")."\");";
?>
```

DCGI

Příklad implementace III

Všechny moderní prohlížeče mají funkci XMLHttpRequest

Bohužel tato funkce je silně závislá na použitém prohlížeči.

- IE podle verze používá
  - new ActiveXObject("Msxml2.XMLHTTP")
  - new ActiveXObject("Microsoft.XMLHTTP")
- Mozila a Safari používají
  - new XMLHttpRequest()
- IceBrowser používá
  - window.createRequest()

Implementace Javascriptu pro IE a Mozilu

```
 <script type="text/javascript">//<![CDATA[
var xmlhttp=false;
/*@cc_on @*/
/*@if (@_jscript_version >= 5)
// JScript gives us Conditional compilation, we can cope with old IE versions.
// and security blocked creation of the objects.
try {
xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
} catch (e) {
try {
xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
} catch (E) {
xmlhttp = false;
}
}
@end @*/
if (!xmlhttp && typeof XMLHttpRequest!='undefined') {
          try {
                    xmlhttp = new XMLHttpRequest();
          } catch (e) {
                    xmlhttp=false;
          }
}
if (!xmlhttp && window.createRequest) {
          try {
                    xmlhttp = window.createRequest();
          } catch (e) {
                    xmlhttp=false;
          }
}
```

Implementace - použití


```
function vyrobDotaz() {
        xmlhttp.open("GET", "ajax_hodiny_server.php",true);
        xmlhttp.onreadystatechange=function() {
                if (xmlhttp.readyState==4) {
                        vypisHodiny(xmlhttp.responseText);
                }
        }
        xmlhttp.send(null)
}

 function vypisHodiny(hodiny_string) {
          document.getElementById("div_hodiny").innerHTML += "<br/>" +
hodiny_string;
        }
```

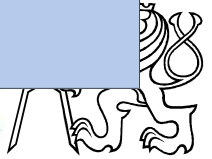
Handler události

Obsluha události odpovědi serveru

ajax_hodiny_server.php

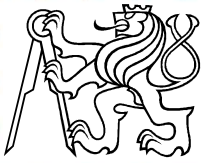
```
<?php
header("Expires: Wed, 23 Dec 1980 00:30:00 GMT");
header("Last-Modified:".gmdate("D, d M Y H:i:s")." GMT");
header("Cache-Control: no-cache, must-revalidate");
header("Pragma: no-cache");

echo date("H:i:s");
?>
```

DCGI



Formát dat

- Formát není definován
- Je třeba sjednotit klientskou a serverovou stranu
- Klient a server tedy není univerzální
- Obvykle se používají např.:
  - pole oddělená čárkou
  - serializovaný Javascript (JSON)
  - nějaká XML formát
  - SOAP
  - pole s pevnou velikostí (např. 20 byte)



Nebezpečí AJAXu

Asynchronní způsob aktualizace stránky

- Není zaručeno, za jak dlouhou dobu server odpoví
- Události NESMÍ mít závislosti – možnost deadlocku nebo nečekaného chování

Příklad asynchronního problému

- async_login_request (data)
- async_get_account_state_request

get_account_state předběhl login

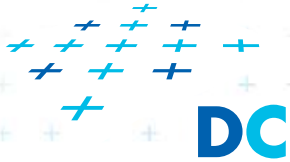
- get_account_state_response (uživatel není zalogován)
- login_response(OK, jste zalogován)

Ajax a session

- Stav aplikace je de-facto udržován v prohlížeči
- „Sessions nejsou potřeba“ ☺

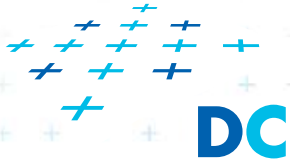
…to je utopie

- vše funguje dobře až do okamžiku, než uživatel znovu načte celou stránku (historie, reload, F5)



Jak na to v praxi – použijme knihovny

- SAJAX
- XAJAX
- AJAXAC
- JPSPAN
- …



XAJAX

- http://xajaxproject.org/
- Obsahuje kninovnu JavaScriptu a PHP
- Jednoduchá manipulace s objekty DOMu pomocí PHP metod
- Uživatel je odstíněn od implementačních detailů javascriptu

Postup

1. vyrobíme PHP funkci, která bude manipulovat s DOMem
   - manipulace pomocí objektu typu xajaxResponse
   - nad xajaxResponse voláme jednotlivé metody manipulace DOM
   - funkce může mít parametry, ty jsou předány z klienta
2. vyrobíme objekt xajax
3. zaregistrujeme funkci u objektu typu xajax
4. zavoláme xajax->processRequest();
5. dále následuje iniciální html stránka
6. v sekci head vypíšeme vygenerovaný javascript
7. použijeme „serverové“ funkce podle libosti voláním funkcí v javascriptu

Ukázka Xajax – aktuální čas

- pro jednoduchost vše uložíme do jednoho jediného php skriptu

```php
require ('xajax_core/xajax.inc.php');
$xajax = new xajax();


function aktualniCas()
{
        $cas = date("H:i:s");
        $objResponse = new xajaxResponse();
        $objResponse->append('div_cas', 'innerHTML', "<br/>$cas");
        return $objResponse;
}

// zaregistrujeme funkci aktualniCas
$reqAktualniCas =& $xajax->registerFunction('aktualniCas');

$xajax->processRequest();

// zde je inicialni stranka
// následuje HTML tránka
```

Xajax objekt

serverová funkce volatelná axynchrnonně z javascriptu

registrace funkce do ajax objektu pod jmenem aktualniCas

spuštění obsluhy – pozná, zda má obsloužit sync či async způsobem

statické html všetně javascript funkcí je na dalším slide

Ukázka Xajax – aktuální čas

```
// ... pokračování z předchozího slide
echo '<?xml version="1.0" encoding="UTF-8"?>';
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
        <title>Xajax aktualni cas</title>
<?php
        $xajax->printJavascript();
?>

</head>
<body>
  <form action="">
        <input type="button" onclick='<?php $reqAktualniCas->printScript();
?>' value="Aktuální čas"/>
  <div id="div_cas"></div>
  </form>
</body>
</html>
```

javascript generovaný PHP knihovnou-

volání registrované funkce

Ukázka Xajax II - našeptávač

Funkce našeptávače

- Jak uživatel píše do textového políčka, je průběžně na server odesílán obsah tohoto pole
- Server text zpracuje a odešle seznam relevantních dat
- Klient reaguje na příjem dat aktualizací nějaké části stránky
- Uživateli se zdá, že mu server našeptává

Implementace

Klient:

- Textové pole
- Javascript reakce na událost napsání textu (keyup)
- Reakce na příjem dat, modifikace DOM

Server:

- Zpracování http request
- Nalezení relevantních dat
- Odeslání události

Klient – HTML + Javascript v Xajax = view

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
    <head>
        <title>naseptavac</title>
        <?php $xajax->printJavascript('./'); ?>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    </head>
    <body>
        <form action="">
            <label for="search">Začněte psát jméno</label><br/>
            <input type="text" onkeyup="xajax_whisper(this.value);"
name="search" id="search"/>

            <br/><label for="whisper">Nalezená jména</label>
            <div id="whisperdiv">
                <select id="whisper" name="whisperselect">
                </select>
            </div>
        </form>
    </body>
</html>
```

Server data z DB = model

```
class Nameday {
    static function getNameday($like, $limit = 10) {
        $query = "
                        SELECT svatek1
                        FROM svatky
                        WHERE svatek1 LIKE ('".addslashes($like)."%')
                        ORDER BY svatek1
                        LIMIT $limit ";
        $result = DB::query($query);
        if (!$result) die ("DB dotaz selhal".mysql_error());

        return $result;
    }

 public static function getNamedayAsSelect ($like, $name, $id, $limit = 10) {
        $to_return = "<select name='".htmlspecialchars($name)."'
id='".htmlspecialchars($id)."'>";
        if (trim($like) != "") {
            $result = self::getNameday($like, $limit);
            while ($row = mysql_fetch_assoc($result)) {

$to_return.="<option>".htmlspecialchars($row['svatek1'])."</option>";
            }
        }
        $to_return.="</select>";
        return $to_return;
    }
}
```

Controller – Xajax kouzlo

```
require ('xajax_core/xajax.inc.php');
require('autoload.php');

function whisper($text)
{
        $objResponse = new xajaxResponse();
        $objResponse->clear("whisperdiv","innerHTML")
       ->append("whisperdiv","innerHTML",
Nameday::getNamedayAsSelect($text,"whisper", "whisper"));
        return $objResponse;
}

$xajax = new xajax();

// registruji naseptavaci funkci
$whisper = $xajax->register(XAJAX_FUNCTION, "whisper");

$xajax->processRequest();

include ('naseptavac_view_bez_cekani.php');


?>
```