



DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

Logika na straně klienta, skriptovací jazyky II

Martin Klíma

Javascript – pokročilé programování

- Vytváření objektů
- Prototypy a dědičnost



V čem se Javascript liší od např. Javy

- Javascript má jiné pojetí objektů
- Neexistuje zde klasická hierarchie tříd
- Třídy vlastně neexistují
- Existují jen instance a tzv. **PROTOTYPY**

Klasika

Class
Instance

Prototypovací jazyk

Class, Instance jedno jest
Prototyp



Vytváření objektů

- Několik možností
 - Objekt Object
 - Constructor funkce
 - Literál objektu
 - Prototyp



Vyrábění instancí pomocí objektu Object

- Třída Object je definovaná v javascriptu defaultně
- Mohu z ní vyrábět instance a těm přidávat vlastnosti dynamicky

```
osoba = new Object();  
osoba.jmeno = "Martin";  
osoba.prijmeni = "Klima";  
osoba.pohlavi = "Muz";  
osoba.bydliste = "Praha";  
  
alert(osoba.jmeno);
```

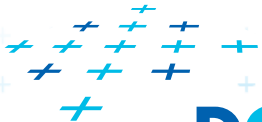
kuk: objects.html



Vyrábění instancí pomocí funkce

- Vypadá jako normální funkce (a lze ji tak použít)
- Privátní, tj. lokální proměnné jsou uvozené slovem **var**
- Ukazatel na instanci třídy je **this**
- Nový objekt vzniká operátorem **new**
- Jedna funkce může být metodou jiné funkce

```
function X {  
  // definice funkce  
}  
  
instance = new X();
```



Funkce – objekt komplexní ukázka

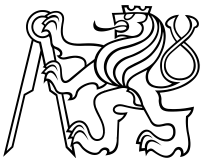
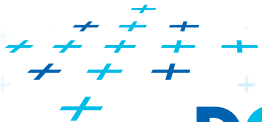
```
// definice tridy osoba
function osoba (jmeno, prijmeni) {
    this.jmeno      = jmeno;
    this.prijmeni   = prijmeni;
    this.pohlavi    = "Muz";
    // definice metody
    this.nastavPrijmeni = nastavPrijmeni;
    // dalsi definice metody
    this.celeJmeno = celeJmeno;
}

// definice metody tridy
// je mimo tuto tridu
function nastavPrijmeni(nove_prijmeni) {
    this.prijmeni = nove_prijmeni;
}

franta = new osoba("Franisek", "Pospisil");
franta.bydliste = "Brno";
franta.nastavPrijmeni("Neruda");
```

Definice třídy, zároveň konstruktor třídy

Přidáme metody, je to ukazatel na jinou funkci



Literály

- Literál je daná hodnota
- Můžeme pomocí nich definovat i objekty
- Objekt je vlastně pole dvojic klíč:hodnota, kde klíč je jméno vlastnosti a hodnota její hodnota nebo

```
franta = {  
    jmeno: "Frantisek",  
    prijmeni: "Pospisil",  
    pohlavi: "Muz",  
    celeJmeno: celeJmeno};
```

Hodnota

Ukazatel na metodu

```
function celeJmeno() {  
    return this.jmeno+" "+this.prijmeni;  
}  
document.write(franta.jmeno);  
document.write("<br>");  
document.write(franta.prijmeni);  
document.write("<br>");  
document.write(franta.celeJmeno());
```

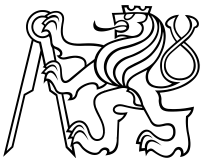
kuk: objects4_initializer.html



Literály – použití anonymních funkcí

```
franta = {  
  jmeno: "Franisek",  
  prijmeni: "Pospisil",  
  pohlavi: "Muz",  
  celeJmeno: function () {  
    return this.jmeno+" "+this.prijmeni;  
  }  
};
```

```
document.write(franta.jmeno);  
document.write("<br>");  
document.write(franta.prijmeni);  
document.write("<br>");  
document.write(franta.celeJmeno());
```



Literály pro pole

- Klasický způsob naplnění pole

```
pismenka = new Array('a', 'b', 'c');
```

- Pole můžeme naplnit při jeho definici pomocí literálů

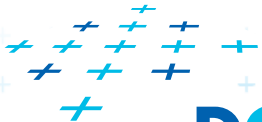
```
pismenka = ['a', 'b', 'c'];
```

```
pismenka = ['a', 'b', 'c', , 'e', ];
```

- Pole můžeme definovat jako děravé

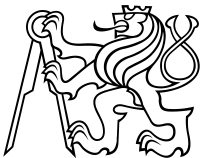
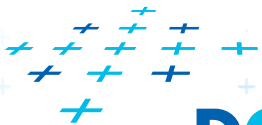
Díra

Díra



Prototypy

- Prototyp je hodnota, ze které se vytváří instance konkrétní třídy
- Každý objekt má vlastnost **prototype**
- Pamatuje si tím, "z čeho vznikl"
- Nastavení nových vlastností prototypu se projeví na všech instancích, které z něho vznikly



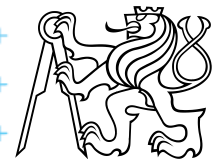
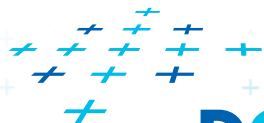
Prototype

```
// definice tridy osoba
function osoba (jmeno, prijmeni) {
    this.jmeno      = jmeno;
    this.prijmeni   = prijmeni;
    // this.pohlavi  = "Muz";
}

franta = new osoba("Frantisek", "Pospisil");
pepa = new osoba("Josef", "Novak");

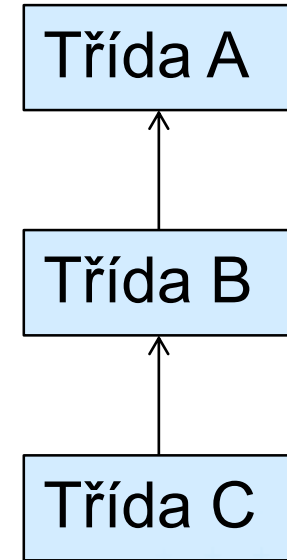
// nyní pridej ke vsem instancim tridy osoba tuto vlastnost
osoba.prototype.pohlavi = "Muz";

document.write(franta.pohlavi);
document.write("<br>");
document.write(pepa.pohlavi);
```



Dědičnost pomocí prototypů

- Chtěl bych "klasickou" dědičnost, tj.
- V javascriptu není rozdíl mezi třídou a instancí
- Finta pomocí prototypů

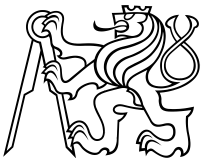
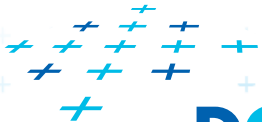


```
function A() {  
    // definice třídy A  
}  
  
function B() {  
    // definice rozšíření B oproti A  
}  
  
B.prototype = new A();
```



Dědičnost pomocí funkcí

```
function A(param1) {  
    // definice třídy A  
    this.param1 = param1;  
}  
  
function B(param1, param2) {  
    this.base = A;  
    this.base(param1);  
    this.param2 = param2;  
}  
  
x = new B(100, 200);  
  
document.write(x.param1 + ", " + x.param2);
```

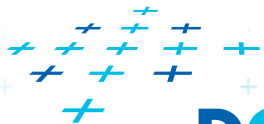


Vzor Singleton v javascriptu

- Singleton je objektový programovací vzor, který umožní udržovat právě jednu instanci objektu v systému
- Problém v javascriptu: nejsou třídy, nejsou třídní proměnné
- Finta: využijeme tzv. anonymní třídu

```
singleton_object = new function Singleton() {  
    // definice tridy zde  
}
```

VZTIKU



Singleton ukázka

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> Singleton </TITLE>
</HEAD>

<script type="text/javascript">

// definice tridy Singleton a její instanciovani
singlePocitadlo = new function Pocitadlo () {
    this.hodnota    = 0;
}

// pouziti
singlePocitadlo.hodnota++;
document.write(singlePocitadlo.hodnota);

</script>
<BODY>

</BODY>
</HTML>
```

Funkci nemusím pojmenovat, ale je to lepší pro ladění

kuk: vzor_singleton.htm



Singleton pokr

■ Přidáme nějaké metody

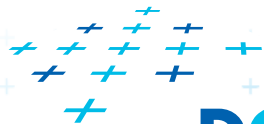
- opět jako anonymní (třídy nebo instance??)

```
// definice tridy Singleton a její instanciovani
singlePocitadlo = new function Pocitadlo () {
    this.hodnota = 0;
    this.inc = function inc () {
        this.hodnota++;
    }

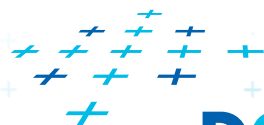
    this.getHodnota = function h () {
        return this.hodnota;
    }
}

// pouziti
singlePocitadlo.inc ();
singlePocitadlo.inc ();
document.write (singlePocitadlo.getHodnota ());
```

Funkci mohu pojmenovat
ale nemusím



AJAX

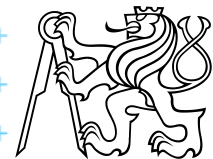


DCGI



AJAX – co to je?

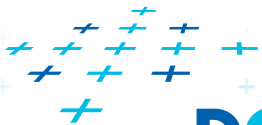
- **Asynchronous Javascript And XML**
- Webový klient komunikuje s webovým serverem asynchronně.
- Výsledkem je jen částečná aktualizace stránky
- Blíží se návrhu klasické desktopové aplikace
- AJAX není nová technologie
- Jde o novou aplikaci existující technologie, resp. o rádobu novinku v souvislosti s pojmem Web 2.0



Ajax

Příklady:

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



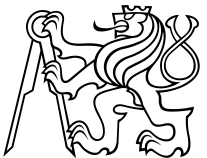
AJAX - jak to funguje

Klasický web

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

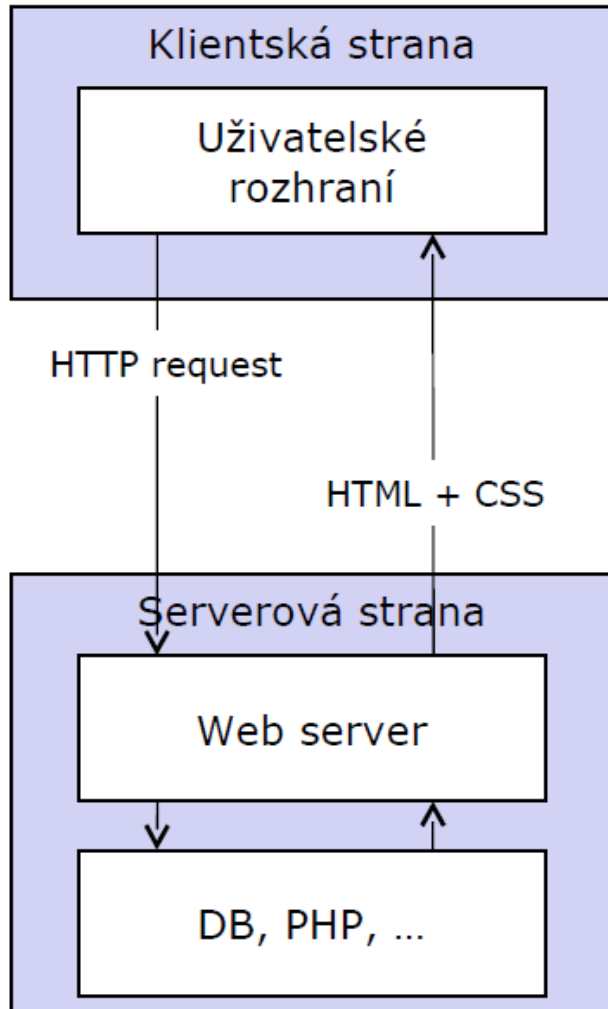
Ajax

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

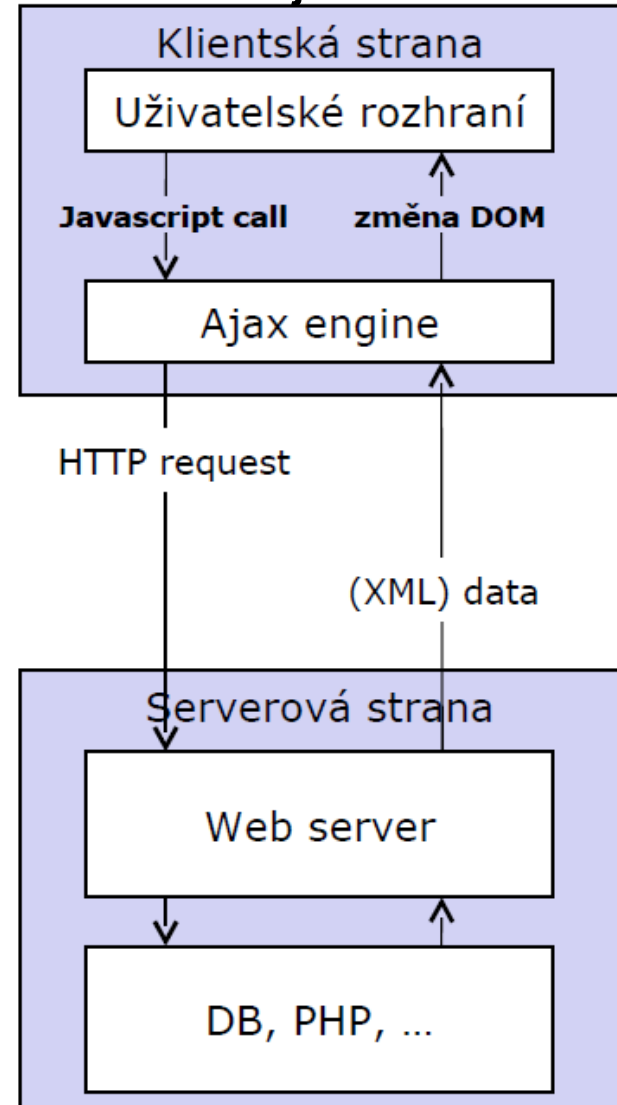


Ajax – jak to funguje II

Klasický web



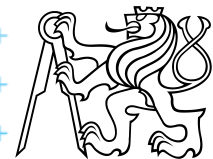
Ajax



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>
```


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 SCRIPT

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">
      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 += "&lt;br/&gt;" +
hodiny_string;
      }
    ]&gt;
  &lt;/script&gt;
&lt;/head&gt;
&lt;body&gt;
  &lt;form action=""&gt;
    &lt;input type="button" value="Kolik je hodin?" onclick="vyrobDotaz()" /&gt;
    &lt;div id="div_hodiny"&gt;&lt;/div&gt;
  &lt;/form&gt;
&lt;/body&gt;
&lt;/html&gt;</pre></div><div data-bbox="339 798 640 833" data-label="Text"><p>skript_generovany_php.php</p></div><div data-bbox="339 850 963 952" data-label="Text"><pre>&lt;?php
echo "vypisHodiny(\"\".date("H:i:s")."\"");";
?&gt;</pre></div><div data-bbox="126 936 218 977" data-label="Page-Footer"><p>DCGI</p></div>
```

Příklad implementace III

Všechny moderní prohlížeče mají funkci
XMLHttpRequest

Bohužel tato funkce je silně závislá na použitém
prohlížeči.

- IE podle verze používá

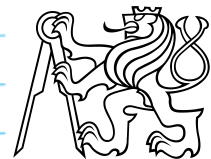
- `new ActiveXObject("Msxml2.XMLHTTP")`
- `new ActiveXObject("Microsoft.XMLHTTP")`

- Mozilla a Safari používají

- `new XMLHttpRequest()`

- IceBrowser používá

- `window.createRequest()`



Implementace Javascriptu pro IE a Mozilu

```
<script type="text/javascript">
var xmlhttp=false;
/*@cc_on @*/
/*@if (@_jscript_version &gt;= 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 &amp;&amp; typeof XMLHttpRequest!='undefined') {
    try {
        xmlhttp = new XMLHttpRequest();
    } catch (e) {
        xmlhttp=false;
    }
}
if (!xmlhttp &amp;&amp; window.createRequest) {
    try {
        xmlhttp = window.createRequest();
    } catch (e) {
        xmlhttp=false;
    }
}
]</pre></div><div data-bbox="125 936 219 977" data-label="Page-Footer"><p>DCGI</p></div><div data-bbox="948 868 982 977" data-label="Image"><img alt="A small, stylized drawing of a person's head and shoulders, possibly a logo or watermark, located in the bottom right corner of the slide."/></div>
```

Implementace - použití

Javascript na kliento

```
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;
}
```

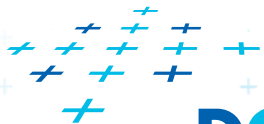
Handler události

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");
?>
```



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)



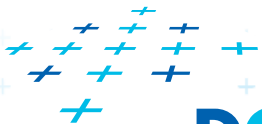
Nebezpečí AJAXu

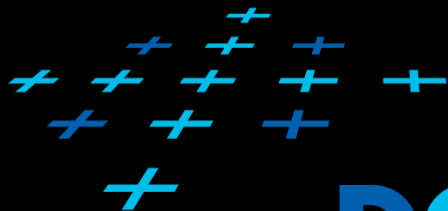
Asynchronní způsob aktualizace stránky

- 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)`





DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

Děkuji za pozornost

Martin Klíma