

# INTERNETOVÉ TECHNOLOGIE – CLIENT SIDE P6

## 2008-11-04

### JAVASCRIPT:

#### JavaScript:

- ✓ Interpreted programming language
- ✓ Program v dokumentu HTML, který **zajišťuje**:
  - Tvorbu aktivního prostředí
  - Kontrolu údajů zadaných uživatelem ve formuláři
  - Generování nových HTML stránek
  - Přístup k vybraným funkcím prohlížeče
  - Zjištění přítomnosti plug-in

#### JavaScript – Java:

- ✓ Zajišťují proveditelný obsah v prohlížečích
- ✓ JavaScript řídí chování a obsah prohlížeče
- ✓ Java pro vývoj samostatných aplikací – tvorba grafiky a síťového zpracování a multithreading
- ✓ Použití Javy na straně klienta se zajišťuje formou tzv. appletů, malých programů, které běží v prohlížeči.
  - Vkládání pomocí značky APPLET
  - pomocí JavaScriptu je možno volat metody Appletu

#### Historie JavaScriptu:

- ✓ **Netscape JavaScript:**
  - JavaScript 1.0 původní název „LiveScript“
  - JavaScript 1.1 – podpora obrázků, polí, Java appletů, plug-inů
  - JavaScript 1.2 – ECMAv1 - Netscape Navigator 4.0
  - JavaScript 1.3 - Netscape Navigator 4.5
  - JavaScript 1.4
  - JavaScript 1.5 – ECMAv3 - Mozilla a Netscape 6
- ✓ **Microsoft Jscript:**
  - JScript 1.0 – „kompatibilita“ s JavaScript 1.0
  - JScript 2.0 – JavaScript 1.1 – IE 3.0
  - JScript 3.0 – JavaScript 1.3 – ECMA v1- IE4
  - JScript 4.0 – JavaScript 1.4
  - JScript 5.0 – ECMAv3 – IE5
  - JScript 5.5 = JavaScript 1.5 = ECMAv3 – IE6
- ✓ **ECMA – [www.ecma.ch](http://www.ecma.ch):**
  - Standardizing Information and Communication Systems
  - ECMAv1 – první standard
  - ECMAv2
  - ECMAv3 – standard (switch, regular expressions, exception handling) = JavaScript 1.5 = JScript 5.5

[ECMA-262](#), ECMAScript Language Specification, 3rd edition

#### Klientský JavaScript:

- ✓ Netscape 6/Mozilla – JavaScript 1.5:
  - Zásadní podpora standardu DOM W3C
- ✓ IE 6 – JScript5.5:
  - Částečná podpora standardu DOM W3C
  - Chybí podpora modelu událostí DOM



### Výhody skriptování:

- ✓ **Snadné použití** – jednoduché kódy programů – vsuvky do HTML simple blocks of code that respond to user actions (event-driven code)
- ✓ **Zpracování na straně klienta**
- ✓ **Rychlý vývoj** – skripty jsou interpretované jazyky
- ✓ **Výkon** – velmi účinný nástroj, ale nižší výkon než kompilovaný kód
- ✓ **Bezpečnostní rizika** – možnosti skriptování jsou omezeny. IE neumožňuje zápis souborů na pevný disk klientského PC.

### Scripty v HTML:

- ✓ **Obecný zápis:**  

```
<SCRIPT LANGUAGE="JavaScript">
<!--
[The actual script commands]
//-->
</SCRIPT>
```
- ✓ **!-- a -->** komentář - pro nižší verze prohlížečů
- ✓ **Odkaz na externí soubor:**  

```
<script src="svatky_cz.js" type="text/javascript"></script>
```

### Notace a názvy:

- ✓ Tečková notace představuje jasný a pohodlný způsob zápisu odkazů na objekty a jejich komponenty. V této konvenci jsou jména objektu, případně jména jemu podřízených objektů, vlastností, událostí, metod a kolekcí odděleny tečkou.
- ✓ Příklad, window.document.linkColor odvolává se na vlastnost linkColor objektu document, který je podřízen objektu window.
- ✓ Vždy se nemusí uvádět celá tečková notace. Příklad document.linkColor odkazuje na vlastnost linkColor objektu document v aktuálním okně.
- ✓ Názvy jsou citlivé na velikost písmen.
  - Pouze malá písmena pro jednoduché názvy (jako je src nebo height)
  - Kombinace malých a velkých písmen pro složené názvy (jako je fileSize nebo fileModifiedDate).

### Skript v HTML:

- ✓ JavaScript se vkládá do HTML mezi tagy <script></script>:  

```
<html>
<head>
<script type='text/javascript'>
var x = 'Hello';
</script>
</head>
<body>
<h1>
<script type='text/javascript'>
document.write(x);
</script>
</h1>
</body>
</html>
```

### Příklad – zápis skriptu:

```
<script language="JavaScript">
var browser = navigator.appName
var verze = parseInt(navigator.appVersion)
if ((browser == "Microsoft Internet Explorer" & verze >= 4) | (browser == "Netscape" & verze >= 4))
{ sirka = screen.width
  vyska = screen.height
```



```
document.write("Rozlišení obrazovky: <b>" + sirka + " x " + vyska + "</b><br>") }  
else document.write("Tyto vlastnosti nejsou Vaším prohlížečem podporovány")  
</script>
```

### Elementární pravidla pro zápis programů:

- ✓ Znaková sada – Unicode
- ✓ Rozlišování velkých a malých písmen
- ✓ Mezerové znaky a ukončení řádku
- ✓ Zápis volitelného středníku ;
- ✓ Rezervovaná slova

### Jádro jazyka:

- ✓ Komentáře
- ✓ Identifikátory
- ✓ Datové typy
- ✓ Proměnné
- ✓ Operátory
- ✓ Příkazy
- ✓ Funkce
- ✓ Objekty

### Komentáře:

- ✓ **Jednořádkový** pomocí //  

```
<script type='text/javascript' >  
...  
// komentář  
...  
</script>
```
- ✓ **Víceřádkový** pomocí /\* ... \*/
  - Nelze vnořovat  

```
<script type='text/javascript' >  
...  
/* komentář  
komentář */  
...  
</script>
```

### Identifikátory:

- ✓ Jsou to jména proměnných, funkcí a vlastností objektů
- ✓ Mají libovolnou délku
- ✓ První znak identifikátoru musí být malé nebo velké písmeno anglické abecedy, ‘\_’ nebo ‘\$’, další znaky mohou být i číslice
- ✓ Case sensitive

### Datové typy:

- ✓ **Primitivní:**
  - Number – číslo
  - String – řetězec
  - Boolean – logická hodnota
- ✓ **Referenční:**
  - Function – reference na funkci
  - Object – reference na objekt
- ✓ **Nedefinovaná hodnota:**
  - Undefined – hodnota proměnné, které ještě nebyla přiřazena hodnota



### Deklarace proměnných:

#### ✓ Příkazem var:

- Var nova\_promenna; Nemusí mít počáteční hodnotu
- Pokud nemá počáteční hodnotu, má přiřazenu hodnotu undefined

#### ✓ Pomocí přiřazení (bez příkazu var):

- Nova\_promenna = hodnota\_nebo\_vyraz;
- Musí mít počáteční hodnotu

#### ✓ Nemají pevně přiřazený typ, mohou obsahovat hodnotu libovolného typu ("loosely typed"):

Var pocet;

pocet = 10;

pocet = „deset“;

#### ✓ Deklarace proměnných musí proběhnout dříve než jejich použití

#### ✓ Globální proměnné:

- Jsou definovány:
  - V hlavním skriptu
  - V bloku přiřazením
- Jsou přístupné uvnitř funkcím

#### ✓ Lokální proměnné:

- Jsou definovány v těle bloku pomocí příkazu var
- Po opuštění bloku se ruší

### Aritmetické operátory:

- ✓ Sčítání +
- ✓ Odčítání -
- ✓ Násobení \*
- ✓ Dělení /
- ✓ Modulo (zbytek po dělení) %
- ✓ Inkrementace ++
- ✓ Dekrementace --

### Přiřazovací operátory:

- ✓ JavaScript nemá speciální přiřazovací operátor
- ✓ Přiřazení =
- ✓ Inkrementace +=
- ✓ Dekrementace -=
- ✓ Násobení \*=
- ✓ Dělení /=
- ✓ Modulo %=

var i = 10

i += 3 // i = i + 3

i \*= 5 // i = i \* 5

### Logické operátory:

- ✓ Obvykle provádějí logické operace nad logickými hodnotami true a false
- ✓ Logický součin &&
- ✓ Logický součet ||
- ✓ Negace !

### Relační operátory:

- ✓ Vrácí logickou hodnotu true, nebo false na základě porovnání hodnot operandů
- ✓ Rovno ==
- ✓ Nerovno !=
- ✓ Větší >
- ✓ Větší nebo rovno >=
- ✓ Menší <



- ✓ Menší nebo rovno <=

### Řetězcové operátory:

- ✓ Mají stejné symboly jako aritmetické symboly pro sčítání
- ✓ Spojování řetězců +, +=
- ✓ Operátory + a += se chápou jako řetězcové, je-li alespoň jeden operand typu řetězec  
var s = "Muž"  
s = s + ", žena"  
s += "a dítě."

### Další operátory:

- ✓ **Podmíněný operátor:**
  - Podmínka ? výraz1 : výraz2
  - Je-li podmínka true provede se výraz1
  - Je-li podmínka false provede se výraz2
- ✓ **Objektové operátory:**
  - Vytvoření nového objektu new
  - Přístup k aktuálnímu objektu this  
var Dnes = new Date();  
<input type='text' name='CODE' onClick='check(this)'>
- ✓ **Operátor typeof:**
  - Slouží ke zjišťování typu operandů
  - Výsledkem je řetězec označení typu
    - "undefined"
    - "number"
    - "boolean"
    - "string"
    - "function"
    - "object"

### Konverze typů:

- ✓ **Konverze na číslo** – provádí se tehdy, pokud se má s jedním nebo dvěma operandy provést matematická operace  
10 \* "aaa" // chyba  
"10" \* "10" // = 100  
10 \* "10" // = 100
- ✓ **Konverze na řetězec** – je-li hodnota předávána jako parametr funkcím, které očekávají řetězcovou hodnotu
- ✓ **Konverze na logickou hodnotu** – provádí se, je-li hodnota operand logické operace, nebo částí podmíněného výrazu

### Přiřazovací příkazy:

- ✓ **Přiřazení hodnoty:**  
a = 10
- ✓ **Matematický výraz:**  
a = ++  
a = a + b
- ✓ **Přiřazení jiné proměnné:**  
a = b + c

### Cykly:

- ✓ **Příkaz while** – opakuje cyklus tak dlouho, dokud je podmínka true  
while (podmínka){  
příkaz;  
};
- ✓ **Příkaz for:**  
for (inicializační\_výraz; podmínka; modifikační\_výraz){



```
příkaz;  
};  
for (i=0; i<25; i++)... ){  
příkaz;  
};
```

- ✓ **Příkaz for...in** – slouží k provedení nějakého příkazu pro všechny vlastnosti objektu  
for (řídící\_proměnná in object){  
příkaz  
};  
for (i in Date)...

### Příkazy řízení běhu:

- ✓ **Větvení programu:**  
if (podmínka){ příkaz }  
else { příkaz }
- ✓ **Příkaz break** – násilně ukončí cyklus a přejde na provádění dalšího příkazu
- ✓ **Příkaz continue** – přeskočí zbytek těla cyklu a pokračuje jeho další iterací

### Příkaz CASE:

- ✓ Provede právě jednu z možností  
switch (proměnná){  
case hodnota : kód; break;  
case hodnota : kód; break;  
....  
};

### Přístupové příkazy

- ✓ **Příkaz with:**
  - Umožňuje přistupovat k vlastnostem a metodám objektu zkráceným způsobem
  - With (objekt){  
příkaz  
};  
with (Math){  
odmocninaPI = sqrt(PI);  
};  
odmocninaPI = Math.sqrt(PI);

### Funkce:

- ✓ Příkazem **function** se vytváří funkce:  
function nazev\_funkce(parametry){  
příkaz; };
- ✓ Příkaz **return** ukončuje funkci, není-li uveden, jedná se o proceduru:  
function nazev\_funkce(parametry){  
příkaz;  
return výsledek; };  
*function soucet(a,b){  
var c = a + b;  
return c;  
};*

### Objekty:

- ✓ Objektový model JavaScriptu je velmi jednoduchý
- ✓ Nemá dědičnost, zapouzdření a polymorfismus
- ✓ Objekty jsou v podstatě kolekce vlastností a metod, které se identifikují jmény nebo čísly
- ✓ JavaScript neumožňuje novou třídu definovat
- ✓ Lze pouze využívat vestavěné třídy



- ✓ Na základě vestavěných tříd a objektů se mohou definovat uživatelské objekty
- ✓ Nový objekt se vytváří pomocí “new” – var novy = new existující\_objekt (parametry)
- ✓ Vlastnosti objektu jsou dostupné pomocí tečkové notace ve tvaru objekt.jmeno\_vlastnosti
- ✓ Všechny objekty mají definovány metody “valueOf” a “toString”, které se používají při konverzi objektu na číslo nebo řetězec

### Odkazy na objekty:

- ✓ Typ objekt **identifikuje**:
  - Objekty jazyka HTML – souvisí se stránkou
  - Objekty prohlížeče
  - Vestavěné objekty
- ✓ Objekty jsou **dostupné** pouze pomocí odkazů
- ✓ Nemůžeme vytvořit **kopii** objektu, ale pouze kopii odkazu na něj

### Vestavěné funkce:

- ✓ Tyto funkce nejsou metodou žádného objektu, ale jsou vestavěny přímo do JavaScriptu
- ✓ **Seznam funkcí**:
  - escape a unescape
  - eval
  - parseFloat
  - parseInt
  - isNaN

### Vestavěné objekty:

- ✓ **Array**:
  - Slouží k vytváření **polí**
  - **Parametrem** je počáteční počet prvků pole
  - Implicitní **vlastností** je „length“, určující počet prvků pole  
var NovePole = new Array(15);  
var PoleTyden = new Array(„Po“, „Út“, „St“, „Čt“, „Pá“, „So“);  
alert(„Pole má“ + NovePole.length + „prvků.“);  
PoleTyden[6] = „Ne“;
- ✓ **Date**:
  - Objekty této třídy **reprezentují** určitý časový okamžik
  - Hodnota času je uložena v milisekundách uplynulých od počátku (1. 1. 1970)
  - Může vzniknout několika **způsoby**:
    - new Date() - vytvoří objekt reprezentující datum a čas vzniku objektu
    - new Date(“měsíc den,rok [hod [:min [:sec]]][GMT]”) - vytvoří objekt reprezentující uvedený čas
    - new Date(milisekundy) - vytvoří objekt reprezentující časový okamžik zadaný v počtu milisekund od počátku
    - new Date(rok, měsíc, den [hod[:min[:sec]]]) - vytvoří objekt reprezentující uvedený čas
  - **Metody Date**:
    - UTC(rok, měsíc, den [hod[:min[:sec]]]) - vrací počet milisekund od počátku k uvedenému času, který je chápán jako greenwickský
    - parse(“měsíc den,rok [hod [:min [:sec]]][GMT]”)
    - Metody objektů třídy Date
      - getDate(), getDay(), getHours(), getMinutes(), getMonth(), getSeconds(), getTime(), getYear()
      - setDate(), setHours(), setMinutes(), setMonth(), setSeconds(), setTime(), setYear()
      - getTimeZoneOffset() - vrátí rozdíl v milisekundách mezi místním časem a GMT
      - toGMTString - převede datum na řetězec vyjadřující greenwickský čas
      - toLocalString() - převede datum na řetězec vyjadřující místní čas
- ✓ **Math**:
  - Tato třída **nemá** žádné **konstruktory**, proto nemůžou vznikat její objekty
  - Slouží k počítání aritmetických výrazů
  - **Konstanty**:



- E – základ přirozených logaritmů
- LN2 – přirozený logaritmus 2
- LN10 – přirozený logaritmus 10
- LOG2E – dvojkový logaritmus čísla E
- LOG10E – dekadický logaritmus čísla E
- PI – Ludolfovo číslo
- SQRT1\_2 – odmocnina z jedné poloviny
- SQRT2 – odmocnina z 2

✓ **String:**

- **Obsahuje** metody pro práci s textovými řetězci
- Lze vytvořit objekt String  

```
var s1 = new String("slovo") // odkaz na objekt
```

```
var s2 = "slovo" // deklarace proměnné
```
- Objekty mají **vlastnost** length, která určuje délku řetězce
- Pokud na řetězcovou proměnnou zavoláme metodu třídy String, konvertuje se proměnná na objekt třídy String, ovšem nemůžou se už přidávat vlastnosti  

```
var delka = "tohle je řetězec".length;
```

✓ **Function:**

- Umožňuje dynamicky vytvářet objekty, které se chovají jako funkce
  - `function([parametry,] tělo_funkce)`
  - Oba příklady jsou ekvivalentní
    - `var scitani = new Function("x", "y", "return x+y");`
    - `function scitani(x, y) {  
 return x + y;  
};`

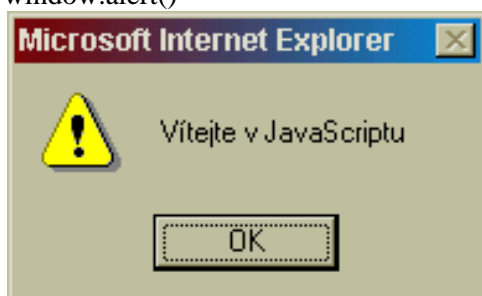
**Metody objektu Window:**

- ✓ Metoda `open(url, jméno_okna, parametry_okna)`
- Otevře nové (případně existující) okno
  - Vrátí referenci na objekt window
  - Url – adresa stránky, která se má v okně zobrazit
  - Jméno\_okna:
    - Název okna, které se má otevřít
    - Jestliže okno s tímto názvem neexistuje, otevře se okno nové
  - Parametry\_okna:
    - Udávají vzhled okna
    - Zadáávají se za sebe a oddělují se čárkou
    - Toolbar, location, directories, status, menubar, scrollbars, resizable, copyhistory, width, height, top, left

```
window.open("file:///C:/", "druhe_okno", "width=300, height=100, menubar=yes");
```
  - Metoda `focus()` – přenesení okna do popředí
  - Metoda `close()`
    - Zavře okno

**Dialogová okna:**

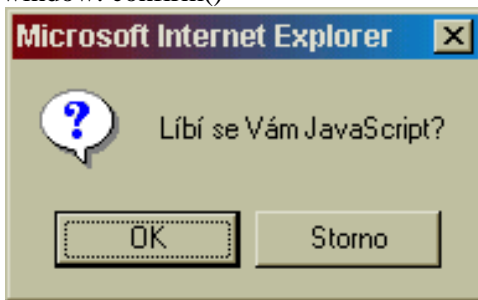
- ✓ `window.alert()`







- ✓ window.confirm()



- ✓ window.prompt()



### Příklad JavaScriptu:

- ✓ Skript vložte do hlavičky stránky:

```
<head>
<title>Untitled</title>
<script language="JavaScript">
<!--
function capitalize() {
//Capitalization of data in field "lastname"
// of form named "form"
document.form.lastname.value=
document.form.lastname.value.toUpperCase();
}
//-->
</script>
</head>
```
- ✓ Parametry vložte do formuláře:

```
<form
action="http://kit.pef.czu.cz/cgi-bin/echo.cgi"
method="GET"
name="form" >
<p>Příjmení
<input type="Text" name="lastname"
onchange="capitalize();" >
</p>
```

### Vyzkoušejte zobrazení času – přidejte skript zobrazující na stránce čas:

```
<!-- text funkce musí být obsažen v hlavičce -->
<SCRIPT language="javascript">
<!--
function currentTime() {
// GIVE CURRENT TIME : ""hh : mm"
var theDate= new Date();
return (theDate.getHours() + " : " + theDate.getMinutes());
}
//-->
```



```
</SCRIPT>
<!-- následující řádky vložte do těla stránky -->
<SCRIPT language="javascript">
<!--
document.write(currentTime() );
//-->
</SCRIPT>
```

### **Použití skriptů – příklady:**

- ✓ Zjištění prohlížeče a rozlišení obrazovky - browser verze.js
- ✓ Změna písma – rozmazaný text - blur text.js
- ✓ Nasvícení nápisu – kuzel svetla.js
- ✓ Létající kurzor – kurzor.js
- ✓ Informace o svátku – svatky-cz.js

### **Otevření okna:**

- ✓ Umístění kurzoru na obrázek
- ✓ Reference

### **Scripty:**

- ✓ <http://java.tatousek.cz/>
- ✓ <http://www.dynamicdrive.com/>
- ✓ <http://wsabstract.com/script/cutindex1.shtml>
- ✓ <http://www.asaisir.com/a-la-page>