

Informační systémy

JavaScript

Martin Trnečka

Katedra informatiky
Univerzita Palackého v Olomouci

Předchozí přednášky

- Jazyk HTML - nástroj pro vytvoření **statické webové stránky**.
- CSS - nástroj pro upravení vzhledu www stránky.
- I přes určitou dynamiku, která přichází s CSS jsou stránky stále statické.
- Potřeba **dynamické webové stránky** (měnící se dle požadavků uživatele, akceptující vstup, zobrazující výstup) - nutnost plnohodnotných webových IS (ale i rozsáhlejší webu).
- Plnohodnotné dynamické stránky - budeme probírat později.
- Dynamické stránky na straně klienta - skriptovací jazyky (JavaScript).

Připomenutí: Klient server architektura. Zatím se pohybujeme stále na straně klienta (omezené možnosti).

Skriptování (na webu)

- Programový kód který nepotřebuje žádný preprocessing (interpretován přímo webovým prohlížečem).
- Přináší nové možnosti a dělá web mnohem dynamičtější (reagování na události, změna obsahu stránky bez reloadu, odesílání a přijímání informací a další).
- Smazává rozdíly mezi klasickým software a www aplikacemi (ne zcela).
- Vykonávání na straně klienta - má své výhody i nevýhody.
- Z předchozího bodu plynou i další omezení, především bezpečnostního charakteru.
- Prakticky jediný (používaný) jazyk: **JavaScript**, JScript, ECMAScript.
- Pod záštitou W3C.

JavaScript

- Dynamicky typovaný, interpretovaný objektový skriptovací jazyk pro www stránky (má i více dalších využití, např. na straně serveru - Node.js, rozšíření pro Adobe Acrobat, Hvězdná brána a další).
- Autor Eich Brendan z již neexistující společnosti Netscape (koupena AOL).
- Poprvé představen v roce 1995, původně měl sloužit jako programovací jazyk (LiveScript) pro Netscape prohlížeči do kterého byla přidána podpora Javy (marketingový tah s Javou nemá téměř nic společného).
- Začátek „temného období“ pro webové stránky, popularita JavaScriptu zapříčinila odklon od původního účelu webu. První experimenty s GUI (strašlivé následky!).
- Microsoft přichází s VBScript (nekompatibilní), později portace JSript pro IE, období největších rozdílů mezi prohlížeči (ActiveX komponenty a problémy s tím spojené). Pokusy o normalizaci ECMAscript.
- **Dnes:** Klid po bouřlivé době, plnohodnotný nástroj pro skriptování na straně uživatele. Velká popularita, celá řada knihoven řešící různou funkcionalitu.

Začlenění JavaScriptu do stránky

1 Externí soubor (přípona .js)

Příklad

```
1 <script src="behavior.js"></script>
2
3 <!-- starsi, v HTML5 defaultni, netreba -->
4 <script type="text/javascript" src="behavior.js"></script>
5
6 <!-- pouze pro externi skripty, asynchroni nactani, HTML5 -->
7 <script src="behavior.js" async></script>
```

2 Přímě do HTML souboru

Příklad

```
1 <script>
2   document.write("Hello World!")
3 </script>
```

Co je tedy lepší?

- Externí skripty se stahují v okamžiku výskytu v HTML kódu.
- Po načtení se skript ihned vykonává dokud neskončí, popřípadě dokud nedojde k chybě.
- Soubor může být umístěn kdekoliv na stránce (totéž platí pro inline JavaScript).
- Pozdější výskyt = pozdější načtení a vykonání skriptu.
- Ne vždy lze umístit kód na konec stránky. Nicméně obvykle to vede ke snížení celkové zátěže (relativně, zátěž je pořád stejná).
- Rozdíl (nijak zásadní záležití na situaci):
 - 1 Externí soubory pomáhají udržet pořádek ve zdrojových kódech (důležité!).
 - 2 Inline kód snižuje zátěž při načítání stránky.

Odbočka: Zátěž při načítání stránky

- Žijeme v 21. století, ale ...
- Obecně je lepší načítat menší počet externích souborů (menší zátěž).
- Velké obrázky.
- Komprese HTML, CSS a JavaScriptu

`http://closure-compiler.appspot.com/`
`http://yui.github.io/yuicompressor/`

- Komfort uživatele, SEO, ...
- Špička ledovce, velice rozsáhlá problematika.
- Pro zájemce: Steve Souders, Even Faster Web Sites, O'Reilly Media, 2009.

Jednoduchý program

Příklad (Jednoduchý JavaScriptový program)

```
1 <script>
2   document.write("<h2>Tabulka faktorialu</h2>");
3   for(i = 1, fact = 1; i < 10; i++, fact *= i) {
4     document.write(i + "! = " + fact);
5     document.write("<br>");
6   }
7 </script>
```

Možnosti JavaScriptu:

- Řízení vzhledu a obsahu dokumentu
- Řízení prohlížeče
- Interakce s formuláři
- Interakce s uživatelem
- Čtení a zápis cookies



Co nelze

- JavaScript nemá žádné grafické schopnosti (s výjimkou silných schopností generovat HTML kód).
- Neumožňuje čtení a zapisování souborů (klientský JavaScript).
- Nepodporuje práci v síti (odesílat a přijímat data samozřejmě lze).
- Již od počátku: nedokáže provádět žádné škodlivé operace (poněkud ne jednoznačné, bezpečnostní mezery).

Tutoriál jazyka JavaScript

- Standardně ASCII nově i Unicode.
- Case Sensitive.
- Ignorují se mezery tabulátory a znaky nového řádku mezi lexikálními elementy.
- Středníky na konci řádku (lze vypustit pokud je každý příkaz na samostatném řádku - zdroj chyb, nejasné situace).
- Poznámky `//`, `/* */`
- **Základní vyhrazená slova:** `break`, `case`, `catch`, `continue`, `default`, `delete`, `do`, `else`, `false`, `finally`, `for`, `function`, `if`, `in`, `instanceof`, `new`, `null`, `return`, `switch`, `this`, `throw`, `true`, `try`, `typeof`, `var`, `void`, `while`, `with`
- Integrovaná automatická správa paměti.

Základní datové typy

- **Čísla.** Všechna čísla jsou reprezentována s pohyblivou desetinou čárkou. 64 bitová reprezentace ($1,7978931348623157 \times 10^{308}$). Lze i čísla v 8 a 16 soustavě. Speciální hodnoty: Infinity, NaN a další. Více na:

http://www.w3schools.com/jsref/jsref_obj_number.asp

Příklad

```
1 var x = 33;  
2 var y = 77;  
3 z = x + 11 - (y * 100);
```

- **Řetězce.** Řetězce lze uvozovat " nebo '.
- **Logické hodnoty** true, false
- **Null**, (undefined proměnná bez hodnoty)
- **Datum**
- Složené datové typy (Objekty) a pole.

Řetězce

Standardní escape sekvence. Operace zřetězení +.

Příklad

```
1 var txt = new String("string");  
2 var txt = "string";
```

Metody:

- []
- charAt()
- trim()
- toLowerCase()
- toUpperCase()

Více na http://www.w3schools.com/jsref/jsref_obj_string.asp

Pole I.

Příklad

```
1 var a = new Array();
2 a[0] = 1;
3 a[1] = "Neco";
4 a[3] = true;
5
6 var a = new Array(10);
7
8 var a = new Array(1, "Neco", true);
9
10 var a = [1, "Neco", true];
```

Vícerozměrná pole matice [x] [y] ;

Pole II.

Alokace paměti jen pro potřebná místa $a[0] = 1$; $a[10000] = 1$; alokuje pouze místo pro dva indexy, ale definuje je!

Příklad

```
1 var k = new Kruh(1, 2, 3);  
2 k[0] = "ma tohle vubec smysl?";
```

Příklad (Délka pole)

```
1 var a = new Array(); // a.length == 0  
2 var a = new Array(10); // a.length == 10  
3 var a = new Array(1,2,3) // a.length == 3  
4 a = [4, 5] // a.length == 2  
5 a[5] = -1 // a.length == 6  
6 a[49] = 0 // a.length == 50
```

Pole III.

Metody pole:

- `join()` - převádí všechny prvky pole do řetězce
- `reverse()`
- `sort()` - bez argumentů = abecední třídění, jinak funkce 2 parametru, která vrací < 0 (správné pořadí), 0 , > 0 (první za druhým)
- `concat()` - přidání prvků na konec pole (rozkládá, ale ne rekurzivně)
- `slice()` - vrací část pole (od do nebo, záporné hodnoty určují konec, jeden argument = od dané pozice)
- `splice()` - obecná metoda pro vkládání a odebírání prvků do polí
- `push()`, `pop()` - práce s polem jako by to byl zásobník.
- `shift()` odebírá, `unshift()` přidává na začátku pole (mění indexy)
- `toString()`

Více na http://www.w3schools.com/jsref/jsref_obj_array.asp

Funkce a objekty

Příklad

```
1 function nadruhou (x) { return x*x; }  
2  
3 // analogie lambda funkce z jazyka LISP  
4 var nadruhou = function (x) { return x*x; }
```

Příklad

```
1 var bod = new Object();  
2 bod.x = 2.3;  
3 bod.y = -1.2;  
4  
5 // nebo  
6 var bod = {x: 2.3, y: -1.2};
```


Proměnné a obor platnosti

Klíčové slovo `var`. Proměnné je třeba deklarovat. Nelze číst nedeklarovanou proměnnou. Přiřazení do nedeklarované proměnné = implicitní deklarace (vzniká globální proměnná).

Příklad (Globální a lokální proměnné)

```
1 var obor = "globalni";
2 function kontrola_oboru() {
3   var obor = "lokalni";
4   document.write(obor);
5 }
6
7 kontrola_oboru (); // vytiskne lokalni
8
9 var obor = "globalni";
10
11 function kontrola_oboru() {
12   obor = "lokalni"; // zmeni hodnotu globalni promenne
13   document.write(obor);
14 }
```

Proměnné a obor platnosti

Pozor není zde obor platnosti na úrovni bloku. **Deklarované proměnné ve funkci jsou deklarované v celé funkci!**

Příklad

```
1 var obor = "globalni";
2 function f() {
3   alert(obor); // undefined
4   var obor = "lokalni"; // definice
5   alert(obor); // lokalni
6 }
7
8 f();
```

Obory platností se řetěží.

Výrazy, operátory a řízení výpočtu

- Většina stejná jako v C, C++ a Java
- Aritmetické operátory, relační operátory, inkrementace, dekrementace, ===, logické operátory, bitové operátory, přiřazující operátory (+=).
- if, else if, else, swich, while, do while, for, break, continue.
- Detailně budou probrány na cvičení.

Vyjimky v JavaScriptu

Příklad

```
1 try {  
2   // kod který může způsobit vyjimku at už chybou nebo příkazem throw  
3 }  
4  
5 catch (e) {  
6   // odchycení vyjimky  
7 }  
8  
9 finally {  
10  // vykonává se vždy po skončení bloku try, nezáleží jakým způsobem byl skončen  
11 }
```

Funkce ještě jednou

Funkce lze v JavaScriptu chápat jako data!

Příklad (Objekt arguments)

```
1 function f(x, y, z) {  
2   if (f.arguments.length !== 3) {  
3     throw new Error("Chybka");  
4   }  
5 }
```

Důsledek. Funkce s proměnným počtem argumentů.

Příklad (Objekt arguments)

```
1 function f(x) {  
2   if (x <= 1) return 1;  
3   return x * arguments.callee(x-1);  
4 }
```

Funkce ještě jednou

Příklad (Vlastnosti funkci)

```
1 jednoznacneCislo. citac = 0;  
2  
3 function jednoznacneCislo() {  
4   return jednoznacneCislo. citac++;  
5 }
```

Funkce lze vnořovat. Funkce mají lexikální obor platnosti (viz. paradigmat programování), tedy běží v oboru své definice nikoliv v oboru svého vykonávání.

Objekty

Příklad (Konstruktor)

```
1 function Obdelnik(s, v) {
2   this.sirka = s;
3   this.vyska = v;
4 }
5
6 obd = new Obdelnik(2, 4);
7
8 /* tezkopadne lze provest pres obecny konstruktor Object */
9 obd = new Object();
10
11 function Obdelnik(s, v) {
12   var obd = new Object();
13   obd.sirka = 10;
14   obd.vyska = 5;
15   return obd;
16 }
17
18 obd = Obdelnik(2, 4);
```

Objekty

Příklad (Metody - dobře promyslete!)

```
1 function vypocitej_plochu() { return this.sirka * this.vyska; }
2 /* metody lze definovat i jako anonymní funkce */
3 function Obdelnik(s, v) {
4     this.sirka = s;
5     this.vyska = v;
6 }
7
8 obd = new Obdelnik(2, 4);
9 obd.plocha = vypocitej_plochu;
10 a = obd.plocha();
```

Příklad (Procházení vlastností objektu)

```
1 var nazvy = "";
2 for(var nazev in obd) { nazvy += nazev + "\n"; }
3 alert(nazvy); // sirka, vyska, plocha
```


Dědičnost

- Prototypová dědičnost. Každý objekt se dědí z **objektu prototypu**.
- Objekt prototypu je na začátku vždy prázdný.
- Všechny vlastnosti v něm definované se přenášejí na potomky.
- Striktně vzato nedochází ke zdědění hodnoty, ale pouze odkazu na hodnotu.
- K dědění dochází pouze při čtení hodnot.
- Obecně používané pro definici společných metod a konstant.

Příklad (Použití prototypu)

```
1 function Kruh(x, y, r) {  
2   this.x = x;  
3   this.y = y;  
4   this.r = r;  
5 }  
6  
7 Kruh.prototype.pi = 3.14159;
```

Dědičnost

Prototyp existuje i k vestavěným třídám (string).

Příklad

```
1 String.prototype.konciNa = function (z) {  
2   return (z == this.charAt(this.length-1));  
3 }
```

Objekty lze chápat i jako asociativní pole operátor `.` lze nahradit `[]`.

Příklad

```
1 objekt.vlastnost ;  
2 objekt["vlastnost"];
```

Objekty obvykle mívají další vlastnosti `constructor`, `toString()`, `hasOwnProperty()`, `propertyIsEnumerable()`, ...

Regulární výrazy

JavaScript obsahuje regulární výrazy (standardní věc - již znáte).

Příklad (Zadání regulárního výrazu)

```
1 var vzor = /s$/;  
2 var vzor = new RegExp("s$");
```

Příznaky: `i` - nerozlišuje velká a malá písmena, `g` - najde všechny shody (nezastaví se při první), `m` - více řádkový režim

Příklad (Použití)

```
1 var text = "JavaScript";  
2 text.search(/ script /i) /* vraci vyskyt nebo -1 nepodporuje g */  
3 text.replace(/ script /gi, "SCRIPT");  
4 "1 plus 2 je rovno 3".match(/\\d+/g); // vraci ["1", "2", "3"]  
5 "123 456 789".split(" ");
```

Více na: http://www.w3schools.com/jsref/jsref_obj_regexp.asp

Řetězce, čísla a další

Příklad (Převod čísla na řetězec)

```
1 var n = 17;  
2 c = n.toString(); // 17  
3 b = n.toString(2); // 100001  
4 o = n.toString(8); // 021  
5 s = n.toString(16); // 0x11
```

Příklad (Převod řetězece na číslo)

```
1 parseInt("3 a neco");  
2 parseFloat("3.14 metru");  
3 parseInt("ahoj"); // vrací NaN
```

Více na: http://www.w3schools.com/jsref/jsref_obj_global.asp

Pozor: eval() !!!

Datum

Příklad (Převod čísla na řetězec)

```
1 var d = new Date();  
2 var d = new Date(milliseconds);  
3 var d = new Date(dateString);  
4 var d = new Date(year, month, day, hours, minutes, seconds, milliseconds );
```

Metody:

- getDate()
- getMonth()
- getFullYear()
- getMilliseconds()
- setMonth()

Více na: http://www.w3schools.com/jsref/jsref_obj_date.asp

Matematika

- Základní matematika dostupná přes objekt Math

Příklad (Převod čísla na řetězec)

```
1 var x = Math.PI; // Returns PI
2 var y = Math.sqrt(16); // Returns the square root of 16
```

- Sinus, cosinus, zaokrouhlování, absolutní hodnota, logaritmus, maximum, minimum a další
- Pouze základní funkce nic pokročilého.

Více na: http://www.w3schools.com/jsref/jsref_obj_math.asp

Předávání hodnotou a odkazem

- 1 Předávání hodnotou - již jsme viděli
- 2 Předávání odkazem

Příklad

```
1 var vanoce = new Date(2014, 12, 24);  
2 var slunovrat = vanoce;  
3 slunovrat.setDate(21);  
4  
5 vanoce.getDate(); // vraci 21
```

Okna prohlížeče a klientský JavaScript

- Objekt `window`. JavaScript umožňuje manipulaci s okny webového prohlížeče prostřednictvím tohoto objektu.
- Spíše přežitek (popup okna).
- Některé funkce již nemají žádný smysl v moderních prohlížečích (například nastavení status baru).
- Jiné se občas hodí (zjištění velikosti okna).

Zajímavé pro nás:

- `window.confirm()`;
- `window.alert()`;
- `window.scrollTo()`;
- `window.history.back()`, `window.history.forward()`
- manipulace s URL (http://www.w3schools.com/jsref/obj_location.asp)

Více na: http://www.w3schools.com/jsref/obj_window.asp

Document Object (HTML DOM)

- Přímá návaznost na HTML stránku.
- HTML stránka se po načtení stává dostupná přes objekt JavaScriptu `document`.
- Lze tedy použít celý aparát JavaScriptu (přesněji `window.document`).

Příklad (Globální a lokální proměnné)

```
1 document.write(<h1>Tohle jsme uz nekde videli</h1>);
```

Co lze:

- Měnit HTML elementy jejich atributy
- Přidávat a odebírat HTML elementy
- Reagovat na vzniklé události.
- Vyvolat události.

Více na: http://www.w3schools.com/js/js_htmlDOM.asp

Document Object Model (DOM)

- Standart W3C.
- DOM, XML DOM, HTML DOM.
- Reprezentuje dokument jako strom.
- Různé úrovně, postupný vývoj, Level 1 - Level 4 (nejnovější).
- Existoval i Level 0 (IE 3).
- Několik nekompatibilních vývojových větví.
- Specifikace Level 4 <http://www.w3.org/TR/dom/> (stále není finální).
- Zjednodušení standardů, specifika pro JavaScriptu a HTML5.
- Většina běžné funkcionality je již podporována. Více informací na: <http://quirksmode.org/dom/core/>.

Elementy

- `document.getElementById()`,
- `document.getElementsByClassName()` (nefunguje v IE ≤ 8),
- `document.getElementsByTagName()`
- Vrací objekt reprezentující nalezený element jinak `null`.
- Obsah elementu lze zpřístupnit přes vlastnost `innerHTML` vráceného objektu.

Příklad

```
1 <p id="jedna">Hello word</p>
2 <div class="blok"></div>
3 <div class="blok"></div>
4
5 <script>
6 document.getElementById('jedna').innerHTML = "Hello";
7
8 var divs = document.getElementsByClassName('blok');
9 for(var i=0; i<divs.length; i++) {
10     divs[i].innerHTML = i;
11 }
12 </script>
```

HTML DOM

Obecně elementy mohou obsahovat další elementy. Použití `innerHTML` je zpřístupní jako text - nelze tedy plnohodnotně použít.

Příklad (Hiearchie DOM)

```
1 <html>
2
3   <head>
4     <title>Ukazkovy dokument</title> <!-- ma jednoho potomka, text node! -->
5   </head>
6
7
8   <body>
9     <h1>Dokument HTML</h1>
10    <p>Toto je <em>jednoduchy</em> dokument</p>
11  </body>
12
13 </html>
```

Typy uzlů v HTML DOM

Element	Konstanta nodeType	nodeType
Element	Node.ELEMENT_NODE	1
Text	Node.TEXT_NODE	3
Document	Node.DOCUMENT_NODE	9
Comment	Node.COMMENT_NODE	8
DocumentFragment	Node.DOCUMENT_FRAGMENT_NODE	11
Attr	Node.ATTRIBUTE_NODE	2

Vlastnosti: `.nodeType` (pro všechny uzly), `.nodeValue` (pro text a atributte node).

Metody pro práci s DOM

Navigace v DOM

- `childNodes[nodenum]`
- `firstChild`
- `lastChild`
- `parentNode`
- `nextSibling`, `previousSibling`

Modifikace DOM

- `appendChild()`
- `removeChild()`
- `replaceChild()`
- `insertBefore()`
- `getAttribute()`, `setAttribute()`
- `removeAttribute()`

Příklad procházení DOM

Příklad

```
1 function projdiDokument(n) {  
2  
3   var pocetElementu = 0;  
4   if(n.nodeType == 1) pocetElementu++;  
5  
6   var potomci = n.childNodes;  
7  
8   for (var i=0; i < potomci.length; i++) {  
9     pocetElementu += projdiDokument(potomci[i]);  
10  }  
11  
12  return pocetElementu;  
13 }  
14  
15 alert (projdiDokument(document));
```

Příklad modifikace DOM

Příklad (Přidání elementu)

```
1 <!DOCTYPE html>
2 <body>
3
4 <div id="div1">
5 <p id="p1">This is a paragraph.</p>
6 <p id="p2">This is another paragraph.</p>
7 </div>
8
9 <script>
10 var para=document.createElement("p");
11 var node=document.createTextNode("This is new.");
12 para.appendChild(node);
13
14 var element=document.getElementById("div1");
15 element.appendChild(para);
16 </script>
17
18 </body>
19 </html>
```


Příklad modifikace DOM

Příklad (Přidání elementu)

```
1 <div id="div1">
2 <p id="p1">This is a paragraph.</p>
3 <p id="p2">This is another paragraph.</p>
4 </div>
5
6 <script>
7 var para=document.createElement("p");
8 var node=document.createTextNode("This is new.");
9 para.appendChild(node);
10
11 var element=document.getElementById("div1");
12 var child=document.getElementById("p1");
13 element.insertBefore(para, child);
14 </script>
```

Příklad modifikace DOM

Příklad (Odstranění elementu)

```
1 <div id="div1">
2 <p id="p1">This is a paragraph.</p>
3 <p id="p2">This is another paragraph.</p>
4 </div>
5
6 <script>
7 var parent=document.getElementById("div1");
8 var child=document.getElementById("p1");
9 parent.removeChild(child);
10 </script>
```

Příklad modifikace DOM

Příklad (Nahrazení elementu)

```
1 <div id="div1">
2 <p id="p1">This is a paragraph.</p>
3 <p id="p2">This is another paragraph.</p>
4 </div>
5
6 <script>
7 var para=document.createElement("p");
8 var node=document.createTextNode("This is new.");
9 para.appendChild(node);
10
11 var parent=document.getElementById("div1");
12 var child=document.getElementById("p1");
13 parent.replaceChild(para, child);
14 </script>
```

Příklad práce s atributy v DOM

Příklad

```
1 <div id="main" data-text="nic">
2   Text
3 </div>
4
5 <script>
6   document.getElementById("main").setAttribute("data-text", "Ahoj");
7   alert (document.getElementById("main").getAttribute("data-text"));
8   document.getElementById("main").setAttribute("style", "border: 1px solid red");
9
10  document.getElementById("main").removeAttribute("data-text");
11  alert (document.getElementById("main").getAttribute("data-text"));
12 </script>
```

JavaScript a CSS

Příklad

```
1 <html>
2 <body>
3
4 <p id="p2">Hello World!</p>
5
6 <script>
7 document.getElementById("p2").style.color="blue";
8 </script>
9
10 <p>The paragraph above was changed by a script.</p>
11
12 </body>
13 </html>
```

Přehled modifikovatelných vlastností:

http://www.w3schools.com/jsref/dom_obj_style.asp

Události

JavaScript umožňuje reagovat na události popřípadě události vykonávat.

Příklad

```
1 <html>
2   <body>
3     <h1 onclick="this.innerHTML='Oops!'">Click on this text!</h1>
4   </body>
5 </html>
```

Základní události: onclick, ondblclick, onmousedown, onmousemove, onmouseover, onmouseout, onmouseup, onkeydown, onkeypress, onkeyup, onblur, onchange, onfocus, onreset, onselect, onsubmit.

Ne všechny události mají význam pro všechny elementy.

Více na: http://www.w3schools.com/jsref/dom_obj_event.asp

Události

Příklad (Přidání ovladače události)

```
1 var b = document.formular.tlacitko ;
2 var staryOvladac = b.onclick ;
3
4 function novyOvladac() { }
5 b.onclick = function() {staryOvladac(); novyOvladac();}
```

Příklad (Registrace ovladače)

```
1 var a = document.getElementById("element");
2 a.addEventListener("mousedown", zpracovaniUdalosti(), true );
3
4 document.formular.submit.addEventListener("submit", function() {overit ();}, false );
```

Poslední parametr `true` (slouží k zachytávání události), `false` (standardní).
Odstranění lze provést pomocí `removeEventListener`. Více na:

http://www.w3schools.com/jsref/dom_obj_event.asp

Příklad

```
1 <p>A script on this page starts this clock:</p>
2 <p id="demo"></p>
3 <button onclick="myStopFunction()">Stop time</button>
4
5 <script>
6   var myVar=setInterval(function() {myTimer()},1000);
7
8   function myTimer()
9   {
10    var d=new Date();
11    var t=d.toLocaleTimeString();
12    document.getElementById("demo").innerHTML=t;
13  }
14
15  function myStopFunction()
16  {
17    clearInterval (myVar);
18  }
19 </script>
```


Cookies

- Data uložena v malém textovém souboru lokálně na počítači.
- Obvykle slouží pro uchování informací o uživateli nebo sezení.
- Nelze na ně spoléhat, je možné je v klientovy vypnout.
- Obecná forma:

```
document.cookie="username=John Doe; expires=Thu, 18 Dec 2013  
12:00:00 GMT";
```

Příklad (Funkce pro nastavení cookies)

```
1 function setCookie(cname, cvalue, exdays) {  
2   var d = new Date();  
3   d.setTime(d.getTime()+(exdays*24*60*60*1000));  
4   var expires = "expires="+d.toGMTString();  
5   document.cookie = cname + "=" + cvalue + "; " + expires;  
6 }
```



Příklad (Práce s cookies)

```
1 function getCookie(cname) {
2   var name = cname + "=";
3   var ca = document.cookie.split(';');
4   for(var i=0; i<ca.length; i++) {
5     var c = ca[i].trim();
6     if(c.indexOf(name)==0) return c.substring(name.length, c.length);
7   }
8   return "";
9 }
10
11 function checkCookie() {
12   var user=getCookie("username");
13   if (user!="") {
14     alert ("Welcome again " + user);
15   }
16   else {
17     user = prompt("Please enter your name:", "");
18     if(user!=" " && user!=null) {
19       setCookie("username", user, 365);
20     }
21   }
22 }
```



Příklad (Praktický příklad: validace formuláře)

```
1 <form name="formular" action="" onsubmit="return validace()" method="post">
2   <input type="text" name="jmeno">
3   <input type="text" name="prijmeni">
4   <input type="submit" value="Odeslat">
5 </form>
6
7 <script>
8 function validace() {
9   var chyba = false;
10  var jmeno = document.formular.jmeno.value; // formular predstavuje kolekci
11  var prijmeni = document.forms["formular"]["prijmeni"].value;
12
13  if (jmeno==null || jmeno=="") {
14    //alert("Jmeno musi byt vyplneno");
15    document.formular.jmeno.focus();
16    chyba = true; }
17
18  if (prijmeni==null || prijmeni=="") {
19    alert("Prijmeni musi byt vyplneno");
20    chyba = true; }
21
22  return !chyba;
23 }
24 </script>
```

Ladění JavaScriptu

- Velké možnosti v prohlížeči Google Chrome.
- Celá řada nástrojů a pluginů.
- Logování do konzole

Příklad (Logování JavaScriptu)

```
1 var myvar = "konec se jiz blizi";  
2  
3 console.log(myvar, "Logged!");  
4 console.info(myvar, "Logged!");  
5 console.warn(myvar, "Logged!");  
6 console.debug(myvar, "Logged!");  
7 console.error(myvar, "Logged!");
```

Paranoidní uživatelé

- Vykonávání JavaScriptu lze v prohlížeči vypnout.
- Pozůstatek z dob „otravných popup oken“ a jiných zvěrstev.
- Fenomén: chyby v JavaScriptu.
- V dnešní době zásadně degraduje webové aplikace.
- Jediná možnost, tag `<noscript>`:


Příklad


```
1 <script>
2   document.write("Hello World!")
3 </script>
4
5 <!-- v HTML5 lze umístit kamkoliv -->
6 <noscript>
7   Your browser does not support JavaScript!
8 </noscript>
```

Důležité pojmy:

- JavaScript, vlastnosti použití a syntaxe.
- Pole, funkce a objekty v JavaScriptu.
- DOM, možnosti JavaScriptu pro práci s DOM, HTML a CSS.

Čtení na doma:

 Flanagan D., JavaScript, The Definitive Guide, 4th Edition, 2001, ISBN 978-0-596-00048-6. (první 3 části).

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

Kniha: JavaScript, The Definitive Guide, 4th Edition

- Český ekvivalent: JavaScript Kompletní průvodce od computer press (překlad 2. vydání). Rozdíl cca 100 stran.
- Excelentní kniha - bible JavaScriptu.
- Referenční příručka.

