Úvod do JavaScript-u

(zdroj: http://w3schools.com)

JavaScript je najpopulárnejší skriptovací jazyk na internete, ktorý funguje vo väčšine webových prehliadačov, ako sú Internet Explorer, Firefox, Chrome, Opera, a Safari.

Čo je JavaScript?

- JavaScript bol navrhnutý pre pridávanie interaktivity do HTML stránok
- JavaScript je **skriptovací jazyk**
- skriptovací jazyk je "odľahčený" programovací jazyk
- JavaScript je zvyčajne **vkladaný priamo do HTML stránok**
- JavaScript je **interpretovaný jazyk** (to znamená, že skripty sú vykonávané prehliadačom bez predchádzajúcej kompilácie)
- Každý môže používať JavaScript bez zakúpenia licencie

Čo vie JavaScript robiť?

- **JavaScript je programovacím nástrojom HTML návrhárov** – zvyčajne HTML autori nie sú programátori, ale JavaScript je skriptovací jazyk s veľmi jednoduchou syntaxou. Skoro každý môže potom vkladať male "kúsky" kódu do jeho HTML stránok.
- **JavaScript môže vložiť dynamický text do HTML stránky** – napr. JavaScript príkaz: *document.write("<h1>" + name + "</h1>");* môže napísať variabilný text do HTML stránky.
- **JavaScript môže zareagovať na udalosti** - JavaScript môže byť vytvorený tak, že sa vykoná, keď sa niečo stane, napr., keď stránka dokončí jej načítanie alebo keď používateľ klikne na nejaký HTML element.
- **JavaScript môže čítať a zapisovať HTML elementy –** JavaScript môže čítať a meniť obsah HTML element.
- **JavaScript môže byť použitý na validovanie dát –** JavaScript môže byť použitý na validovanie formy dát predtým, než sú predložené na server. To uchráni server pred ich ďalším spracovaním.
- **JavaScript môže byť použitý na detekovanie návštevníkovho prehliadača** – a potom v závislosti na návštevníkovom prehliadači môže JavaScript načítať inú stránku navrhnutú pre jeho prehliadač
- **JavaScript môže byť použitý na vytváranie cookies** - JavaScript môže byť použitý na uloženie a získanie informácií na návštevníkov počítač a o návštevníkovom počítači.

JavaScript = ECMAScript

JavaScript je implementáciou ECMAScript language štandardu. ECMA-262 je oficiálnym JavaScript štandardom.

JavaScript bol objavený Brendanom Eichom v Netscape (v Navigatore 2.0) a bol schválený vo všetkých prehliadačoch v roku 1996.

ECMA štandard (nazývaný ECMAScript-262) bol schválený ako medzinárodná ISO norma (ISO/IEC 16262) v roku 1998.

Jeho vývoj stále pokračuje.

Ako pužiť JavaScript

HTML <script> značka sa používa na vloženie JavaScript-u do HTML stránky.

Písanie do HTML dokumentu
Nasledujúci príklad **zapíše <p> element s aktuálnym dátumom** do HTML dokumentu:

```
<html>
<body>
<h1>My First Web Page</h1>
<script type="text/javascript">
    document.write("<p>" + Date() + "</p>");
</script>
</body>
</html>
```

Modifikovanie HTML elementov
Nasledujúci príklad **zapíše aktuálny dátum do existujúceho <p> elementu** v HTML dokumente:

```
<html>
<body>
<h1>My First Web Page</h1>
<p id="demo"></p>
<script type="text/javascript">
    document.getElementById("demo").innerHTML=Date();
</script>
</body>
</html>
```

Vysvetlenie príkladov
Pre vloženie JavaScript-u do HTML stránky používane <script> značku. *type* atribút tejto značky používame na definovanie skriptovacieho jazyka.
Takže značky <script type="text/javascript"> a </script> povedia prehliadaču, kde JavaScript začína a kde končí.

```
<html>
<body>
<script type="text/javascript">
    ...
</script>
</body>
</html>
```

Riadky medzi značkami <script> a </script> obsahujú **JavaScript, ktorý je vykonaný webovým prehliadačom**. V prípade nasledujúceho kódu prehliadač **zapíše aktuálny dátum do existujúceho elementu s id="demo"** v HTML dokumente.

```
<html>
<body>
<script type="text/javascript">
    document.getElementById("demo").innerHTML=Date();
</script>
</body>
</html>
```

Pozn.: Bez značiek <script> a </script> by prehliadač zapísal
"document.getElementById("demo").innerHTML=Date();"
ako čistý text do webovej stránky.

Kde vložiť JavaScript

JavaScripts môže byť vložený do <body> a <head> sekcií HTML stránky.

JavaScript v <body> sekcii
Nasledujúci kód **zapíše aktuálny dátum do existujúceho <p> elementu s id="demo"** HTML dokumentu, **keď je tento dokument načítaný**.

```
<html>
<body>
<h1>My First Web Page</h1>
<p id="demo"></p>
<script type="text/javascript">
    document.getElementById("demo").innerHTML=Date();
</script>
</body>
</html>
```

JavaScript je umiestnený dole na stránke pre zaručenie toho, aby nebol vykonaný predtým, ako je <p> element vytvorený.

JavaScript funkcie a udalosti
JavaScripty budú v HTML stránke **vykonané pri jej načítaní**. To však nemusí byť to, čo chceme.
Niekedy chceme **vykonať JavaScript** vtedy, keď sa **vyskytne udalosť** v HTML stránke, napr. keď používateľ klikne na tlačidlo nachádzajúce sa v tejto HTML stránke. Keď sa toto udeje, vtedy môžeme mať **skript vložený v tele** nejakej obslužnej udalostnej **funkcie** spojenej s touto udalosťou. Keď sa takáto udalosť vyskytne, je zavolaná s ňou spojená funkcia a v jej tele je vykonaný požadovaný skript.

JavaScript v <head> sekcii

Nasledujúci kód volá funkciu **displayDate**() keď používateľ klikne na tlačidlo s popisom *Display Date*

```
<html>
<head>
<script type="text/javascript">
function displayDate()
{
   document.getElementById("demo").innerHTML=Date();
}
</script>
</head>

<body>
<h1>My First Web Page</h1>
<p id="demo"></p>
<button type="button" onclick="displayDate()">Display Date</button>
</body>
</html>
```

Scripty v <head> a <body> sekciách

Môžeme umiestniť nelimitovaný počet skriptov v našom HTML dokumente a **môžeme ich mať v <head> a <body> sekciách súčasne**.

Obecná prax je vložiť všetky funkcie do <head> sekcie alebo dole v stránke do <body> sekcie. Pri takomto spôsobe umiestnenia sú všetky na jednom mieste a nemiešajú sa s obsahom stránky.

Používanie externých JavaScript-ov

JavaScript môže byť tiež umiestnený v externom súbore.

Externé JavaScript súbory často obsahujú kód, ktorý môže byť použitý viacerými rozdielnymi webovými stránkami.

Externé JavaScript súbory majú príponu *.js*.

Pozn.: Externý JavaScript nemôže obsahovať <script> a </script> tagy!

Keď používame externý skript, musíme na *.js* súbor s ním ukázať v "src" atribúte <script> tagu:

```
<html>
<head>
<script type="text/javascript" src="xxx.js"></script>
</head>

<body>
</body>
</html>
```

JavaScript príkazy

JavaScript je **sekvencia príkazov**, **ktoré budú vykonané prehliadačom**.

JavaScript je citlivý na výšku písma (Case Sensitive)
Na rozdiel od HTML JavaScript je citlivý na výšku písma (príkazy, názvy premenných, objektov a funkcií).

JavaScript príkazy
JavaScript príkaz je príkaz pre prehliadač, ktorému povie, čo má vykonať.
Tento JavaScript príkaz povie prehliadaču, aby napísal "Hello Dolly" do webovej stránky:

```
document.write("Hello Dolly");
```

Pozn.: Používanie **;** na konci príkazu umožňuje napísať viacero príkazov na jednom riadku.

JavaScript kód
alebo len JavaScript je sekvencia JavaScript príkazov.
Každý príkaz je vykonaný prehliadačom v takom poradí, v akom sú za sebou napísané.
Nasledujúci kód napíše nadpis a dva paragrafy (odstavce) do webovej stránky:

```
<script type="text/javascript">
    document.write("<h1>This is a heading</h1>");
    document.write("<p>This is a paragraph.</p>");
    document.write("<p>This is another paragraph.</p>");
</script>
```

JavaScript bloky
JavaScript príkazy môžu byť spolu zoskupené do blokov.
Bloky začínajú otváracou zloženou zátvorkou **{** a končia zatváracou zloženou zátvorkou **}**.
Účelom bloku je vykonať sekvenciu príkazov spolu.
Nasledujúci kód napíše nadpis a dva paragrafy (odstavce) do webovej stránky:

```
<script type="text/javascript">
{
    document.write("<h1>This is a heading</h1>");
    document.write("<p>This is a paragraph.</p>");
    document.write("<p>This is another paragraph.</p>");
}
</script>
```

Uvedený príklad nie je veľmi užitočný, len demoštruje použitie bloku. Blok zvyčajne zoskupuje príkazy vo funkcii alebo v podmienke, kde by mala byť **vykonaná skupina príkazov**, **ak je splnená podmienka**.

JavaScript funkcie

JavaScript funkcia bude vykonaná pomocou udalosti alebo pomocou jej volania.

JavaScript funkcie

Ak chceme uchrániť prehliadač od vykonávania skriptov pri načítaní webovej stránky, tak musíme vložiť naše skripty do funkcie.
Funkcia obsahuje kód, ktorý bude vykonaný pomocou udalosti alebo pomocou volania funkcie.
Funkciu môžeme volať z ľubovoľného miesta stránky (alebo dokonca aj z iných stránok, ak je funkcia vložená v externom *.js* súbore).
Funkcie môžu byť definované v <head> a v <body> sekcii dokumentu. Avšak, ak chceme aby bola funkcia načítaná/zavedená prehliadačom predtým, ako je zavolaná, mali by sme ju prezieravo vložiť do <head> sekcie.

Ako definovať funkciu

Syntax

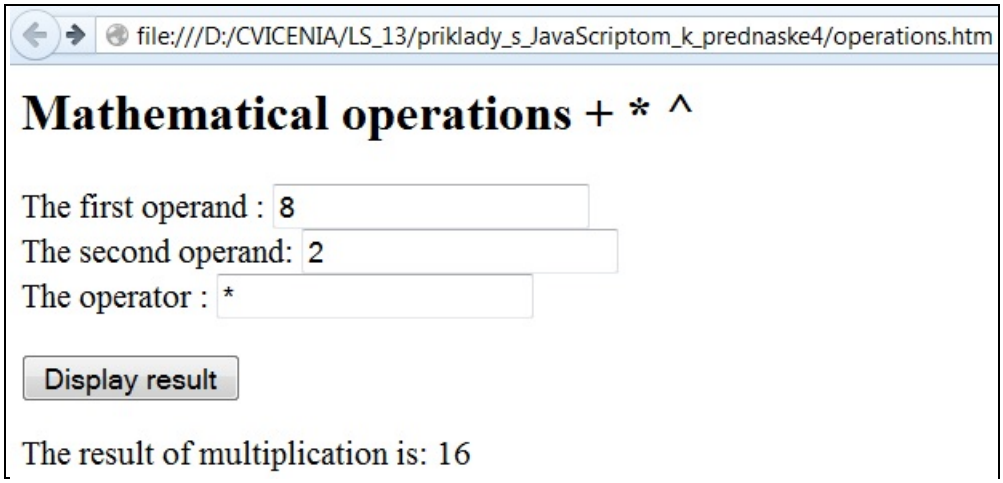
```
function functionname(var1,var2,...,varX)
{
    nejaký kód
}
```

Parametre var1, var2, atď sú premenné alebo hodnoty odvzdané do funkcie. Zátvorky { a } definujú začiatok a koniec tela funkcie (definície funkcie).
Pozn.: Aj funkcia bez parametrov musí mať v hlavičke prázdne okrúhle () zátvorky.

Príklad použitia JavaScript funkcií (zdroj tohto odstavca: autor dokumentu)

Zdanie úlohy:
Pomocou **JavaScript funkcií** vytvorte jednoduchú webovú aplikáciu s používateľským rozhraním, ktorá si cez textové polia načíta od používateľa hodnoty dvoch operandov a vykoná s nimi zvolenú operáciu, ktorú používateľ zvolí vložením jej operátora do príslušného textového poľa. Ak používateľ klikne na tlačidlo s popisom *Display result*, tak webová aplikácia otestuje ním vložený operátor a ak je povoleným operátorom (jeden z **+ * ^**(mocnina)), tak vykoná požadovanú operáciu s príslušnými operandmi. Výsledok operácie aplikácia napíše do odstavca pod tlačidlo s popisom *Display result*. Ak používateľ vložil nepovolený operátor, tak mu to aplikácia oznámi v dialógovom okne a vymenuje mu povolené operátory.

Výrez z okna webového prehliadača so spustenou webovou aplikáciou, ktorá vykonala a zobrazila výsledky požadovanej matematickej operácie:



Kód možného riešenia:

```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html;charset=ISO-8859-1">
<script type="text/javascript">
function sum()
{
 //Number objekt je objektova obalka pre primitivne numericke hodnoty.
 //syntax "var num = new Number(value);". value parameter, ak je to mozne, je konvertovany na cislo
 //vytvorenie Number objektov pomocou konstruktora Number()
 var x = new Number(document.getElementById("first_op").value);
 var y = new Number(document.getElementById("second_op").value);
 return x+y;
}

function product()
{
 var x = new Number(document.getElementById("first_op").value);
 var y = new Number(document.getElementById("second_op").value);
 return x*y;
}

function power()
{
 var x = new Number(document.getElementById("first_op").value);
 var y = new Number(document.getElementById("second_op").value);
 //Pomocou Math objektu mozme vykonat matematicke ulohy. Math objekt nema konstruktor (je to
 //vstavany objekt, vacsina webovych prehliadacov ho pozna). Vsetky vlastnosti alebo metody objektu Math
 //mozu byt volane ihned, bez vytvorenia tohto objektu.
 return Math.pow(x,y);
}
```

```
//*********************************************************************
function Find_operation()
{
 var op=document.getElementById("operator").value;

 if (op=="+")
 {
  var result=sum();
  var string="The result of addition is: " + result;
  document.write(string); //tu prepiseme cele okno LEN obsahom string-u
 }
 else if (op=="*")
 {
  var result=product();
  var string="The result of multiplication is: " + result;
  //tu vlozime retazec string dovnutra HTML elementu s id="par1", u nas je to paragraf (odstavec)
  document.getElementById("par1").innerHTML=string;
 }
 else if (op=="^")
 {
  var result=power();
  var string="The result of the first operand to be the power of the second operand is: " + result;
  document.getElementById("par1").innerHTML=string;
 }
 else
 {
  var string = "Vlozili ste nekorektny operator " + op + " !!!" + "\n" + "\n" + "Mozte vkladat LEN tieto
operatory: + * ^";
  alert(string);
  //nastavenie fokusu (aktivneho kurzora) do textoveho pola s id="operator"
  document.getElementById("operator").focus();
 }
}
</script>
</head>

<body>

<h2>Mathematical operations + * ^</h2>
<form>
 The first operand : <input type="text" id="first_op"/><br>
 The second operand: <input type="text" id="second_op"/><br>
 The operator      : <input type="text" id="operator"/><br>
</form>

<script type="text/javascript">
 //nastavenie fokusu (aktivneho kurzora) do textoveho pola s id="first_op"
 document.getElementById("first_op").focus();
</script>
```

```
<!-- vytvorenie tlacidla s popisom "Display results" a priradenie volania funkcie 'Find_operation' k jeho udalosti 'onclick' (jedno kliknutie) -->
<button type="button" onclick="Find_operation()">Display result</button>

<p id="par1">paragraph - it will be written by a result of a selected and executed mathematical operation (except +)</p>

</body>
</html>
```

Krátky popis riešenia:
Keď používateľ tejto webovej aplikácie klikne na tlačidlo s popisom *Display result*, vyvolá tým na tomto tlačidle udalosť *onclick*. Keďže tlačidlo má do udalostného atribútu *onclick* vložené volanie funkcie **Find_operation**(), tak sa táto JavaScript funkcia zavolá. V jej tele sa testuje znak vložený používateľom do textového poľa s id="operator". Ak používateľ vložil do tohto poľa jeden z povolených operátorov, tak sa z funkcie **Find_operation**() zavolá ďalšia, vloženému operátoru zodpovedajúca, funkcia. Môže to byť jedna z funkcií **sum**(),**product**() alebo **power**(). Príslušnému vloženému operátoru zodpovedajúca zavolaná funkcia vráti volajúcej funkcii **Find_operation**() návratovú hodnotu, čo je výsledok príslušnej matematickej operácie. Funkcia **Find_operation**() si túto návratovú hodnotu uloží do premennej *result* a s príslušným textom ju zobrazí vo forme reťazca dovnútra HTML elementu s id="par1", v našej webovej aplikácii je to paragraf (odstavec).
Ak však používateľ vložil do textového poľa s id="operator" nepovolený operátor, tak funkcia **Find_operation**() to zistí a do dialógového okna používateľovi napíše vložený nepovolený operátor a zoznam povolených operátorov, pričom žiadnu matematickú operáciu s týmto nepovoleným operátorom nevykoná. Pokiaľ používateľ nevloží korektný operátor, tak webová aplikácia bude stále zobrazovať uvedené dialógové okno.

Zadanie pre študentov:
Pokúste sa webovú aplikáciu rozšíriť o operátory **- / 2**(operátor druhej odmocniny prvého operanda) a k nim prislúchajúce funkcie a testy (delenie nulou, odmocnina záporného čísla atď.)

Úvod do JavaScript objektov

Objektovo-orientované programovanie
JavaScript je objektovo-orientovaný programovací (OOP) jazyk. OOP jazyky nám umožňujú definovať svoje vlastné objekty a urobiť svoje vlastné typy premenných.
My sa však budeme viac zaoberať **vstavanými JavaScript objektmi**, ako sa dajú používať.
Objekty sú v JavaScript-e používané ako špeciálny druh dát. **Objekty majú vlastnosti a metódy**.

Vlastnosti
sú hodnoty asociované s objektom.
V nasledujúcom príklade používamme vlastnosť ***length*** objektu *String* pre vrátenie počtu znakov v reťazci *txt*:

```
<script type="text/javascript">
var txt="Hello World!";
   document.write(txt.length);
</script>
```

Výstup uvedeného kódu bude nasledovný:

12

Metódy
sú akcie, ktoré môžu byť vykonané na objektoch.
V nasledujúcom príklade používamme metódu ***toUpperCase()*** objektu *String* prte zobrazenie textu v reťazci *str* veľkými písmenami:

```
<script type="text/javascript">
var str="Hello world!";
   document.write(str.toUpperCase());
</script>
```

Výstup uvedeného kódu bude nasledovný:

HELLO WORLD!

JavaScript Array objekt

---

sa používa na uloženie viacerých hodnôt v jednej premennej. **JavaScript Array objekt má** definované **vlastnosti a metódy**, ktoré môžu byť preň volané a používané. Kompletný zoznam vlastností a metód JavaScript Array objektu aj s ich popisom sa nachádza na www.w3schools.com.

Čo je pole?
Pole je špeciálna premenná, ktorá môže obsahovať viac ako jednu premennú.
Zoznam položiek, napr. značiek áut, uchovávajúci tieto značky v jednotlivých premenných, môže vyzerať nasledovne:

```
var car1="Saab";
var car2="Volvo";
var car3="BMW";
```

Ak by sme chceli prechádzať týmito premennými a hľadať špecifickú značku, pri počte 3 by sme to možno zvládli, ale čo ak by takýchto premenných bolo 300?
Vtedy by bolo najlepším riešením použitie poľa, ktoré by obsahovalo všetkých 300 premenných pod jedným názvom. K jednotlivým hodnotám by sme mohli pristupovať odkazovaním sa na prvky poľa cez jeho ID.

Vytvorenie poľa
Pole môžeme definovať tromi spôsobmi.
Nasledujúci kód vytvorí objekt ***Array*** pomenovaný ***myCars***:

1.

```
var myCars=new Array(); //v zátvorke môže byť určená veľkosť poľa celočíselným argumentom
myCars[0]="Saab";
myCars[1]="Volvo";
myCars[2]="BMW";
```

2.

```
var myCars=new Array("Saab","Volvo","BMW"); //zhutnená definícia rovnakého poľa
```

3.

```
var myCars=["Saab","Volvo","BMW"]; //pole literálov (vymenovaných prvkov)
```

Pozn.: Ak špecifikujeme čísla alebo true/false hodnoty vo vnútri poľa, potom typ premennej poľa bude číselný alebo Boolean (logický) namiesto String.

Prístup k prvkom poľa
je možný cez názov poľa spolu s príslušným indexom prvku (v hranatej zátvorke), ku ktorému pristupujeme. Prvky poľa sú indexované od 0.
Nasledujúcim JavaScript príkazom

```
document.write(myCars[0]);
```

získame takýto výstup:

Saab

Modifikovanie hodnôt v poli
je možné cez prístup k jeho jednotlivým prvkom pomocou jeho indexov, napr. týmto príkazom:

```
myCars[0]="Opel";
```

Vložíme reťazec "Opel" do 0-tého prvku poľa **myCars**.
Potom nasledujúci JavaScript príkaz

```
document.write(myCars[0]);
```

získa takýto výstup:

Opel

Window objekt

---

window objekt representuje **otvorené okno vo webovom prehliadači**.
Ak dokument obsahuje rámy (<frame> alebo <iframe> značky), tak prehliadač vytvorí jeden window objekt pre HTML dokument a jeden prídavný window objekt pre každý rám.
Pozn.: Neexistuje žiaden verejný štandard pre window objekt, ale väčšina prehliadačov tento objekt podporuje.

Vlastnosti window objektu (z w3schools.com v angličtine)

Property	Description
closed	Returns a Boolean value indicating whether a window has been closed or not
defaultStatus	Sets or returns the default text in the statusbar of a window
document	Returns the Document object for the window (See Document object)
frames	Returns an array of all the frames (including iframes) in the current window
history	Returns the History object for the window (See History object)
innerHeight	Sets or returns the the inner height of a window's content area
innerWidth	Sets or returns the the inner width of a window's content area
length	Returns the number of frames (including iframes) in a window
location	Returns the Location object for the window (See Location object)
name	Sets or returns the name of a window
navigator	Returns the Navigator object for the window (See Navigator object)
opener	Returns a reference to the window that created the window
outerHeight	Sets or returns the outer height of a window, including toolbars/scrollbars
outerWidth	Sets or returns the outer width of a window, including toolbars/scrollbars
pageXOffset	Returns the pixels the current document has been scrolled (horizontally) from the upper left corner of the window
pageYOffset	Returns the pixels the current document has been scrolled (vertically) from the upper left corner of the window
parent	Returns the parent window of the current window
screen	Returns the Screen object for the window (See Screen object)
screenLeft	Returns the x coordinate of the window relative to the screen
screenTop	Returns the y coordinate of the window relative to the screen
screenX	Returns the x coordinate of the window relative to the screen
screenY	Returns the y coordinate of the window relative to the screen
self	Returns the current window
status	Sets the text in the statusbar of a window
top	Returns the topmost browser window

Metódy window objektu (z w3schools.com v angličtine)

Method	Description
alert()	Displays an alert box with a message and an OK button

blur()	Removes focus from the current window
clearInterval()	Clears a timer set with setInterval()
clearTimeout()	Clears a timer set with setTimeout()
close()	Closes the current window
confirm()	Displays a dialog box with a message and an OK and a Cancel button
createPopup()	Creates a pop-up window
focus()	Sets focus to the current window
moveBy()	Moves a window relative to its current position
moveTo()	Moves a window to the specified position
open()	Opens a new browser window
print()	Prints the content of the current window
prompt()	Displays a dialog box that prompts the visitor for input
resizeBy()	Resizes the window by the specified pixels
resizeTo()	Resizes the window to the specified width and height
scroll()	
scrollBy()	Scrolls the content by the specified number of pixels
scrollTo()	Scrolls the content to the specified coordinates
setInterval()	Calls a function or evaluates an expression at specified intervals (in milliseconds)
setTimeout()	Calls a function or evaluates an expression after a specified number of milliseconds

HTML DOM Document object

Document objekt
Každý HTML dokument načítaný do okna webového prehliadača sa stáva **Document objektom.**
Document objekt poskytuje prístup zo skriptu **ku všetkým HTML elementom** na web stránke.
Document objekt je tiež časťou Window objektu a môžeme k nemu pristúpiť cez vlastnosť tohto Window objektu **window.document**.
Pozn.: Document objekt môže tiež používať vlastnosti a metódy Node objektu.

W3C: W3C Standard.
Vlastnosti Document objektu (z w3schools.com v angličtine)

Property	Description	W3C
anchors	Returns a collection of all the anchors in the document	Yes
applets	Returns a collection of all the applets in the document	Yes
body	Returns the body element of the document	Yes
cookie	Returns all name/value pairs of cookies in the document	Yes
documentMode	Returns the mode used by the browser to render the document	No
domain	Returns the domain name of the server that loaded the document	Yes

forms	Returns a collection of all the forms in the document	Yes
images	Returns a collection of all the images in the document	Yes
lastModified	Returns the date and time the document was last modified	No
links	Returns a collection of all the links in the document	Yes
readyState	Returns the (loading) status of the document	No
referrer	Returns the URL of the document that loaded the current document	Yes
title	Sets or returns the title of the document	Yes
URL	Returns the full URL of the document	Yes

Metódy Document objektu (z w3schools.com v angličtine)

Method	**Description**	**W3C**
close()	Closes the output stream previously opened with document.open()	Yes
getElementsByName()	Accesses all elements with a specified name	Yes
open()	Opens an output stream to collect the output from document.write() or document.writeln()	Yes
write()	Writes HTML expressions or JavaScript code to a document	Yes
writeln()	Same as write(), but adds a newline character after each statement	Yes