Úvod

Tvorba internetových stránok je veľmi populárna činnosť najmä pre začínajúcich programátorov. Postupným zdokonaľovaním týchto schopnosti a obohacovaním o ďalšie programovacie jazyky je zároveň skvelou vstupnou bránou do sveta webových aplikácií. Pre pokročilú tvorbu internetových stránok je potrebné poznať viacero jazykov, pričom sa odporúča nasledujúce poradie: HTML resp. XHTML, CSS, JavaScript, DOM, BOM, XML, AJAX, PHP, SQL (napr. MySQL, MS SQL, Oracle, PostgreSQL), alebo grafika cez Silverlight či FLASH.

Skôr ako sa niekto začne učiť programovať, je vhodné mu odporučiť, vytvárať si poznámky a príklady z toho čo sa už naučil. Vďaka tejto činnosti je možné veľmi ľahko vybudovať kvalitnú znalostnú databázu. Časom si programátori pre zefektívnenie práce vytvárajú vlastné knižnice, ktoré obsahujú najčastejšie používané objekty, funkcie a premenné. Pri tvorbe dokumentov XHTML s jazykmi CSS, JavaScript a DOM je dobré každý dokument, najmä čoraz zložitejší, otestovať vo viacerých internetových prehliadačoch, minimálne však v Opere, Firefoxe, Internet Exploreri, Safari a Google Chrome. Testovaním sa programátor presvedčí, či každý internetový prehliadač interpretuje napísaný kód správne. Správnosť zobrazovania internetových stránok by mal zabezpečiť aj dodržaný štandard a odporúčania. Najviac práce s identickým vzhľadom pre rôzne internetové prehliadače budú mať programátori najmä pri používajú kaskádových štýlov (CSS) DOM a BOM. Umením dobrého programátora je naprogramovať identicky zobrazovaný web pre každý z najpoužívanejších internetových prehliadačov. Súčasťou internetových stránok sú často aj skripty, ktoré robia weby viac interaktívnymi. Medzi najčastejšie používané skriptovacie jazyky v dokumentoch XHTML patria JavaScript a JScript. V minulosti bolo možné vídať aj VBscript (Visual Basic script). Neskôr pre dynamické generovanie stránok, pokročilejšie funkcie a prácu s databázami je nutné prejsť na niektorý z ďalších programovacích jazykov, ako sú napríklad PHP či ASP.

Táto kniha bola vytvorená za účelom rýchleho naučenia a pochopenie jazyka JavaScript podľa štandardu ECMAScript. Za týmto účelom bola vytvorená prehľadná štruktúra, aby sa v nej každý vedel rýchlo zorientovať. Pre elimináciu chýb boli všetky uvedené príklady kontrolované aplikáciou Firebug a chybovou konzolou internetového prehliadača Firefox. Kniha je určená pre tých, ktorí ešte neprogramovali, začínajúcich programátorov, ale aj ako prehľadná a podrobná príručka pre skúsených programátorov. Svojim spracovaním je vhodná aj ako učebná pomôcka do škôl a vzdelávacích inštitúcií.

Frameworky v JavaScripte

Programovací jazyk JavaScript môže pre rozšírenie svojej funkcionality pracovať s frameworkami, ktoré rozvíjajú funkcionalitu DOM a BOM. Medzi tieto frameworky patria jQuery, MooTools, Prototype, Dojo, ExtJS, Mochikit, GooXdoo, SmartClient, ASP.NET 2.0 AJAX (Atlas), Echo, Rico a ďalšie. Tieto frameworky sa medzi sebou líšia syntaxou, rýchlosťou spracovania kódu, kompatibilitou s internetovými prehliadačmi a ich podporou. Medzi najznámejšie frameworky patria jQuery, MooTools a Prototype. Frameworky, ako externé knižnice, ponúkajú pre rýchlejšiu tvorbu programov vytvorené funkcie a objekty. Autor tak nemusí navrhovať a testovať vlastné algoritmy, ale môže využiť už osvedčené postupy a metódy, ktoré navyše sú kompatibilné aj s internetovými prehliadačmi. Framework je externý súbor s jazykom JavaScript, ktorý sa pripája prostredníctvom elementu **<script>** k dokumentu XHTML[7]. Po pripojení Frameworku k dokumentu je možné využívať funkcie a objekty Frameworku v programovom kóde JavaScriptu.

Príklad pripojenia frameworku prototype k dokumentu XHTML. Element ***<script>*** *sa môže umiestniť aj do elementu* ***<head>****.*

```
<script type="text/javascript" src="prototype-1.6.0.2.js"></script>
```

Poznámka

> Pre testovanie rýchlosti frameworkov MooTools, JQuery, Prototype, YUI a Dojo je možné použiť internetovú stránku http://mootools.net/slickspeed/#. V pravom hornom rohu sa nachádza tlačidlo na odštartovanie rýchlostného testu jednotlivých frameworkov.

JScript

Kým JavaScript je implementáciou ECMAScriptu spoločnosťou Mozilla, JScript je implementáciou ECMAScriptu Microsoftom. Spoločnosť Microsoft v súčasnosti disponuje podporou dvoch typov JScriptu. Prvý, vo verzii 5.8, označovaný ako Microsoft Windows Script Technologie, je zahrnutý do internetového prehliadača Internet Explorer. Druhý, s označením JScript.Net, je vytvorený pre prostredie Microsoft .NET s rozšírenými funkciami oproti štandardu ECMA 262, pričom obsahuje spätnú kompatibilitu s klasickým JScriptom. JScript .NET je v súčasnosti k dispozícii vo verzii 10.0. Používa sa pre aplikácie bežiace na serveroch s frameworkom .NET. Viac informácií o jazyku JScript sa nachádza na internetových stránkach **http://msdn.microsoft.com/en-us/library/72bd815a.aspx** a **http://msdn.microsoft.com/en-us/libra ry/d1et7k7c(VS.85).aspx**.

---

[7] Jazyk XHTML je nástupcom jazyka HTML 4.01. Nástupcom jazyka XHTML je jazyk HTML 5.0, keďže verzia XHTML 2.0 nebola IT komunitou akceptovaná. Verzia HTML 5.0 vychádza z jazyka XHTML a nie HTML. Dôvodom je striktné dodržiavanie pravidiel jazyka XML, z ktorého vychádza jazyk HTML. Práve nedostatočná podpora pravidiel jazyka XML v jazyku HTML bola dôvodom vzniku jazyka XHTML.

Vlastnosť case sensitive

Skriptovací jazyk JavaScript je case sensitive - citlivý na veľké a malé písmená. Nesprávny zápis príkazu, metódy, vlastnosti, objektu, funkcie alebo akéhokoľvek iného identifikátora spôsobí chybu programu. To znamená, že napríklad zápisy `premenna` a `Premenna` sú dva rôzne identifikátory a preto ich nie je možné zameniť.

Príklad č.11. *Platný zápis identifikátora ako rezervované slovo `true`. Keďže názov identifikátora začína veľkým písmenom, identifikátor nie je rezervovaným slovom (vlastnosť case sensitive).*

```
var True = 10;
document.write(True);
```

Rezervované slová JavaScriptu

V rámci JavaScriptu vznikli rezervované slová, ktoré nie je možné používať ako názvy identifikátorov okrem svojho účelu, na ktorý sú určené. Rezervované slová sa nazývajú aj kľúčovými slovami. Reprezentujú slová zo slovníka JavaScriptu s vopred definovanými spôsobmi ich spracovania.

Rezervované slová jazyka JavaScript tvoria nasledujúce dve skupiny:

- **základné rezervované slová**
- **rezervované slová do budúcnosti**

Medzi základné rezervované slová jazyka JavaScript podľa ECMA-262 3rd Edition patria:

```
break       else        new         var         case        finally
return      void        catch       for         switch      while
continue    function    this        with        default     if
throw       delete      in          try         do          instanceof
typeof
```

Medzi rezervované slová do budúcnosti podľa ECMA-262 3rd Edition patria:

```
abstract    enum        int         short       Boolean     export
interface   static      byte        extends     long        super
char        final       native      class       float       synchronized
package     throws      const       goto        private     transient
debugger    implements  protected   volatile    double      import
public
```

Okrem uvedených slov sú rezervovanými slovami aj:

```
null  true   false
```

Príklad č.19. *Použitie premennej typu `Boolean`.*

```
var x = true;
if (x == 1) document.write("OK");
```

Príklad č.20. *Použitie premennej typu `Boolean`.*

```
var x = true;
if (x) document.write("OK");
```

Príklad č.21. *Použitia hodnoty typu `Boolean` v podmienkach.*

```
if (3 > 4) document.write("Pravda, 3 > 4 <br/>");
if (3 < 4) document.write("Pravda, 3 < 4 <br/>");
if ((3 < 4)==true) document.write("Pravda, 3 < 4 <br/>");
if ((3 > 4)==false) document.write("Nepravda, 3 > 4 <br/>");
```

Obrázok: *Zobrazenie príkladu v internetovom prehliadači.*

Pravda, 3 < 4
Pravda, 3 < 4
Nepravda, 3 > 4

Ak premenná má priradenú primitívnu hodnotu `Boolean`, pri matematickej operácii s číslami sa hodnoty `true` a `false` prevádzajú na čísla. Hodnota `true` má číslo 1 a hodnota `false` číslo 0.

Príklad č.22. *Konverzia hodnôt `true` a `false` na čísla 1 a 0.*

```
var p1 = true, p2 = false;
document.write(" p1: " + p1 + " je " + typeof(p1) + "<br/> p2: " + p2 + " je " +
typeof(p2) + "<br/>");
p1 = p1 + 5;
p2 = p2 + 5;
document.write(" p1: " + p1 + " je " + typeof(p1) + "<br/> p2: " + p2 + " je " +
typeof(p2) + "<br/>");
```

Obrázok: *Zobrazenie príkladu v internetovom prehliadači.*

p1: true je boolean
p2: false je boolean
p1: 6 je number
p2: 5 je number

Príklad č.23. *Konverzia výrazov na stav `true`.*

```
var Objekt = new Array();
if ("") document.write("true <br/>");              // false
if ("Reťazec") document.write("true <br/>");       // true
if (10) document.write("true <br/>");              // true
if (Objekt) document.write("true <br/>");          // true
```

Aritmetické operátory

Aritmetické operátory umožňujú vykonávať matematické operácie.

Operátor	*Význam*
+	sčíta dve čísla, c = a + b.
-	odčíta druhé číslo od prvého, c = a - b.
/	delí prvé číslo druhým, c = a / b.
*****	násobí čísla, c = a * b.
%	delí prvé číslo druhým a vracia celočíselný zvyšok po delení, c = a % b.
++	a++, zvýši číslo, najprv použije hodnotu a potom ju zvýši o 1 (postfixové zvýšenie hodnoty).
++	++a, zvýši číslo, najprv hodnotu zvýši o 1 potom ju použije (prefixové zvýšenie hodnoty).
--	a--, zníži číslo, najprv použije hodnotu a potom ju zníži o 1 (postfixové zníženie hodnoty).
--	--a, zníži číslo, najprv hodnotu zníži o 1 potom ju použije (prefixové zníženie hodnoty).
-	ako záporné znamienko hodnoty.

Poznámka

Operátory ++ a – sa môžu používať ako prefixové a postfixové. Ich spracovanie sa rôzni.

Príklad č.33. *Použitie aritmetických operátorov.*

```
var a = 1;
var b = 1;
var x = 5;
var y = 4;
var z;
document.write('Hodnota A:' + a++ + ' Hodnota B:' + ++b + '<br/>');
document.write('Hodnota A:' + a + ' Hodnota B:' + b +'<br/>');
document.write(x + '-' + y + '=' + (x - y) + '<br/>');
document.write(x + '+' + y + '=' + (x + y) + '<br/>');
document.write(x + '/' + y + '=' + (x / y) + '<br/>');
document.write(x + '*' + y + '=' + (x * y) + '<br/>');
document.write(x + '%' + y + '=' + (x % y) + '<br/>');
document.write(-x + '-(' + -y + ')=' + (-x - (-y)) + '<br/>');
```

Príklad č.69. *Použitie operátora* ***this*** *s operátorom* `.` *pre vytvorenie vlastností objektu* `Auto`*.*

```
function VytvorAuto(param1, param2) {
this.typ = param1;
this.rok = param2;
}
var Auto = new VytvorAuto("Mazda", "2009");
document.write(Auto.typ + " " + Auto.rok);
```

Obrázok: *Zobrazenie príkladu v internetovom prehliadači.*

Mazda 2009

Príklad č.70. *Použitie operátora* ***this*** *s operátorom* `[]` *pre vytvorenie vlastností objektu* `Auto`*.*

```
function VytvorAuto(param1, param2) {
this["typ"] = param1;
this["rok"] = param2;
}
var Auto = new VytvorAuto("Mazda", "2009");
document.write(Auto.typ + " " + Auto.rok);
```

Príklad č.71. *Použitie operátora* ***this*** *v metóde.*

```
function vytvor(param1, param2) {
this.typ = param1;
this.rok = param2;
this.metoda = function() {return this.rok - 2000 + 10 };
}
var Auto = new vytvor("Mazda", "2009");
document.write(Auto.typ + " " + Auto.rok + " " + Auto.metoda() );
```

Obrázok: *Zobrazenie príkladu v internetovom prehliadači.*

Mazda 2009 19

Operátor typeof

Operátor **typeof** vracia typ overovanej hodnoty. Prostredníctvom operátora **typeof** je možné testovať podporu vlastností a metód v internetových prehliadačoch. Operátor vráti jednu z hodnôt `undefined`, `object`, `boolean`, `number`, `string` alebo `function`. V prípade objektov E4X XML a E4X XMLList vráti hodnotu `xml`. Parametrom môžu byť objekty, premenné alebo primitívne hodnoty. Keďže `typeof` je operátor a nie funkcia, operand nemusí byť uvedený v oblých zátvorkách.

Formáty zápisu

```
typeof parameter
```

```
typeof(parameter)
```

Vlastnosť index

Vlastnosť **index** reprezentuje vnútorný číselný ukazovateľ objektu. Index najčastejšie reprezentuje pozíciu prvku v objekte. Číslovanie indexu začína nulou. Prostredníctvom vlastnosti **length** je možné získať poslednú číselnú hodnotu prvku v objekte. Tá sa získa zápisom `length-1`, keďže vlastnosť **length** pracuje s číslami od 1 v prípade existencie prvku. Vlastnosť **index** je možné použiť len na vybrané objekty JavaScriptu a DOM.

Príklady zápisu

```
pole[1] = prvok;
```

```
var vysledok = pole[1];
```

Príklad zápisu v DOM.

```
var vysledok = document.forms[1].id;
```

Vlastnosť length

Vlastnosť **length** vráti počet prvkov objektu. V prípade reťazca vráti počet znakov reťazca.

Príklady zápisu

```
var vysledok = Pole.length;
```

```
var vysledok = "Mazda".length;
```

Príklad č.116. *Použitie vlastnosti **length** na pole a reťazce.*

```
var Auto = new Array("Mazda", "323");
document.write("Pole Auto má " + Auto.length + " prvky.<br/>");
var text = "Text.";
document.write("Premenná text obsahuje reťazec " + text + ", ktorý má " +
text.length + " znakov.<br/>");
document.write("Prázdny reťazec má " + "".length + " znakov.<br/>");
```

Obrázok: *Zobrazenie príkladu v internetovom prehliadači.*

Pole Auto má 2 prvky.
Premenná text obsahuje reťazec Text., ktorý má 5 znakov.
Prázdny reťazec má 0 znakov.

Obrázok: *Zobrazenie príkladu v internetovom prehliadači.*

Veľkosť poľa: 4
Výpis poľa: A,B,C,D
Zmazaný prvok: D
Veľkosť poľa: 3
Výpis poľa: A,B,C
Zmazaný prvok: A
Veľkosť poľa: 2
Výpis poľa: B,C

Metóda unshift()

Metóda **unshift()**, vkladá prvky na začiatok poľa, pričom vráti novú hodnotu dĺžky poľa. Metóda pracuje s tým istým poľom, nevytvára nové pole.

Formáty zápisu

```
Pole.unshift(prvok1, ..., prvokN);
var vysledok = Pole.unshift(prvok1, ..., prvokN);
```

Príklad č.135. *Použitie metódy* ***unshift()****.*

```
var Pole = new Array ("A", "B", "C", "D");
var navratovaHodnota = Pole.unshift(1, 2, 3);
document.write("Vrátená hodnota: " + navratovaHodnota + "<br/>");
document.write("Veľkosť poľa: " + Pole.length + "<br/>");
document.write("Výpis poľa: " + Pole.toString() + "<br/>");
```

Obrázok: *Zobrazenie príkladu v internetovom prehliadači.*

Vrátená hodnota: 7
Veľkosť poľa: 7
Výpis poľa: 1,2,3,A,B,C,D

Metóda push()

Metóda **push()** doplní nové prvky na koniec poľa a vráti novú dĺžku poľa. Na rozdiel od metódy **concat()** nevytvára nové pole, ale pracuje s tým istým polom.

Formát zápisu

```
Pole.push(prvok1, ..., prvokN);
```

Metóda substr()

Metóda **substr()** vráti subreťazec z reťazca definovaný začiatkom a počtom znakov od začiatku.

Formát zápisu

```
var vysledok = Retazec.substr(zaciatok[, dlzka]);
```

Ak sa neuvedie nepovinný argument `dlzka`, bude vyňatý reťazec od argumentu `zaciatok` až po koniec reťazca. Argument `dlzka` reprezentuje počet znakov od argumentu `zaciatok` a nie pozíciu indexu v reťazci.

Príklad č.265. *Použitie metódy **substr()**.*

```
var retazec = "Textový reťazec na testovanie.";
var vysledok = retazec.substr(5, 10);
document.write(vysledok + "<br/>");
vysledok = retazec.substr(5);
document.write(vysledok + "<br/>");
vysledok = retazec.substr(5, 17);
document.write(vysledok + "<br/>");
```

Obrázok: *Zobrazenie príkladu v internetovom prehliadači.*

vý reťazec
vý reťazec na testovanie.
vý reťazec na tes

Metóda substring()

Metóda **substring()** vráti subreťazec z reťazca definovaný indexmi `zaciatok` a `koniec`. Vyhľadávanie môže byť vykonané zľava doprava alebo aj sprava doľava v závislosti od hodnoty indexov.

Formát zápisu

```
var vysledok = Retazec.substring(zaciatok[, koniec]);
```

Ak argument `koniec` nie je uvedený, je vyňatý reťazec od argumentu `zaciatok` až po koniec reťazca.

Príklad č.266. *Použitie metódy **substring()**.*

```
var retazec = "Textový reťazec na testovanie.";
var vysledok = retazec.substring(5, 10);
document.write(vysledok + "<br/>");
vysledok = retazec.substring(5);
document.write(vysledok + "<br/>");
vysledok = retazec.substring(5, 17);
document.write(vysledok + "<br/>");
vysledok = retazec.substring(10, 2);
```

Metóda fixed()

Metóda **fixed()** reprezentuje element `<tt>` z jazyka HTML.

Formát zápisu

```
Retazec.fixed();
```

Metóda fontcolor()

Metóda **fontcolor()** reprezentuje zápis `<font color="color">` z jazyka HTML.

Formát zápisu

```
Retazec.fontcolor(farba);
```

Metóda fontsize()

Metóda **fontsize()** reprezentuje zápis `<font size="size">` z jazyka HTML.

Formát zápisu

```
Retazec.fontsize(velkost);
```

Argument `velkost` je celé číslo od 1 do 7.

Metóda italics()

Metóda **italics()** reprezentuje element `<i>` z jazyka XHTML.

Formát zápisu

```
Retazec.italics();
```

Metóda link()

Metóda **link()** reprezentuje zápis `<a href="uri">` z jazyka HTML.

Formát zápisu

```
Retazec.link(URI);
```

Metóda small()

Metóda **small()** reprezentuje element `<small>` z jazyka XHTML.