

Tvorba internetových stránek

*pomocí HTML, CSS
a JavaScriptu*

Martin Domes

Nakladatelství a vydavatelství

Computer Media®

Vzdělávání, které baví
www.c-media.cz

Obsah

VYSVĚTLIVKY K PRVKŮM POUŽITÝM V KNIZE	10
JAK PRACOVAT S KNIHOU? JAK JE KNIHA ČLENĚNA?	10
SLOVO AUTORA	11
ÚVODEM	14
HTML A INTERNET	15
Co to je HTML?	15
Oficiální verze HTML	15
Z historie HTML	16
PŘÍPRAVA	17
Čím začít?	17
Pravidla tvorby webu	19
<i>Jednoduchost a přehlednost</i>	<i>19</i>
<i>Správné zobrazení</i>	<i>19</i>
<i>První stránka webu</i>	<i>19</i>
<i>Texty na webu</i>	<i>20</i>
<i>Odkazy na webu</i>	<i>20</i>
<i>Aktuálnost a zpětná vazba</i>	<i>20</i>
<i>Správný zdrojový kód a syntaxe jazyka</i>	<i>21</i>
Webdesign	21
<i>Rozlišení obrazovky</i>	<i>21</i>
<i>Grafická podoba stránek</i>	<i>22</i>
<i>Grafika a obrázky na webu</i>	<i>23</i>
<i>Zpracování grafiky pro web</i>	<i>24</i>
Webové prohlížeče	24
<i>Internet Explorer</i>	<i>25</i>
<i>Mozilla a Mozilla Firefox</i>	<i>26</i>
<i>Opera</i>	<i>26</i>
<i>Zobrazení v různých prohlížečích</i>	<i>26</i>
V čem psát zdrojový kód	26
<i>Strukturní editory</i>	<i>27</i>
<i>WYSIWYG editory</i>	<i>27</i>
HTML	30
Zdrojový kód	30
<i>Párové a nepárové značky</i>	<i>31</i>
<i>Zápis parametrů značek</i>	<i>32</i>

p.př. 1
H3 { margin: 5px;
OL { list-style-type:
LI { margin-top: 10px;
<TH class="px

Tvorba internetových stránek pomocí HTML, CSS a JavaScriptu

Komentář ve zdrojovém kódu	32
Velikosti v HTML	33
Označení umístění objektů	33
Barvy na webu	34
Základní struktura HTML dokumentu	35
Hlavička HTML dokumentu	36
Tělo HTML dokumentu	37
Úprava textu a jeho formátování	39
Úprava textu	40
Formátování a zarovnávání textu	43
Rozdělení dokumentu horizontální linkou	47
Úprava písma	49
Speciální znaky a mezera	53
Seznamy	55
Neuspořádaný a uspořádaný seznam	56
Definiční výčet	59
Obrázky	60
Odkazy	63
Hypertextové odkazy	64
Jmenné odkazy	67
Obrázek jako odkaz	69
Další možnosti odkazů	70
Jak odkazy používat	71
Rámy	72
Tvorba struktury ráků	72
Vkládané rámy	79
Rámy a odkazy	81
Negativa spojená s rámy	85
Tabulky	86
Struktura tabulky	87
Základní definice tabulky – TABLE	88
Tělo tabulky a jeho součásti – TR a TD	89
Tvorba tabulek	91
Úprava rámečku tabulky	96
Rozdělení tabulky – THEAD, TFOOT a TBODY	100
Další možnosti tabulky	102
Problematika tabulek	105
Formuláře	110
Definice formuláře – FORM	110



Vstupní prvek – INPUT	111
Textová pole – TEXTAREA	115
Výběrová nabídka – SELECT	118
Skupina polí – FIELDSET	121
Tlačítko – BUTTON	122
Další možnosti formulářů	123
Závěrem	126
KASKÁDOVÉ STYLY – CSS	128
Co jsou to kaskádové styly?	128
Standardy CSS	129
Vývoj kaskádových stylů	129
CSS a webové prohlížeče	130
Syntaxe CSS	131
Základní syntaxe	131
Zápis selektorů	132
Přívlastkový selektor	134
Třídy a identifikátory	135
Pseudotřídy a pseudoelementy	136
Vlastnosti	136
Poznámky ve stylech	138
Jednotky, barvy a umístění objektů v CSS	138
Jednotky v CSS	138
Zápis barev v CSS	139
Označení umístění objektů	140
Kam zapisovat styly	140
Zápis stylů v těle dokumentu HTML	141
Zápis stylů stylpisem v hlavičce HTML dokumentu	141
Zápis stylů v externím souboru	142
Pozadí a barvy	142
Vlastnosti písma	148
Vlastnosti textu	152
Prázdná mezera	158
Rozměry, okraje, rámečky	161
Rozměry	162
Okraje	165
Rámeček	169
Vizuální formátování	177
Rodičovský a sourozenecký element	177
Oddíly	178

Blokové elementy a oddíly	178
Řádkové elementy a oddíly	179
Zabíhající oddíly	179
Povaha elementu	180
Shrnutí	182
Umísťování	182
Typy umístění	183
Obsahový oddíl a zobrazovací pole	183
Umístění do pozice	184
Plovoucí oddíly	189
Třetí rozměr – vytvoření vrstev	191
Vzájemné vztahy vlastností DISPLAY, POSITION a FLOAT	193
Seznamy	193
Pseudotřídy a pseudoelementy CSS	198
Pseudotřídy odkazů	199
Dynamické pseudotřídy	199
Pseudotřída prvního potomka	202
Pseudoelementy textu	203
Automaticky generovaný obsah	205
Pseudoelementy BEFORE a AFTER	206
Generování obsahu	206
Automatické číslování	210
Tabulky v CSS	213
Základní vzhled tabulky	213
Vizuální podoba tabulky	216
Rámeček	219
Vizuální efekty	224
Přetečení obsahu	224
Odstřížení elementu	226
Viditelnost elementu	228
Uživatelské rozhraní	230
Kurzor	230
Integrace systémových barev	233
Integrace systémových fontů	237
Tvorba dynamického obrysu	238
JAVASCRIPT	244
Co to je JavaScript?	244
Kam zapisovat skript	245
Syntaxe jazyka	246

<i>Komentář ve skriptu</i>	247
<i>Datové typy</i>	247
<i>Operátory a operace s výrazy</i>	248
<i>Proměnné</i>	250
<i>Příkazy</i>	251
<i>Podmínky</i>	252
<i>Cykly</i>	253
<i>Funkce</i>	255
<i>Objekty</i>	256
<i>Pole</i>	258
<i>Události</i>	260
<i>Neviditelné znaky, přerušování řádku a rozdělení kódu</i>	261
<i>Speciální znaky v JavaScriptu</i>	262
<i>Barvy a označení umístění objektu</i>	262
Programování s okny	263
Práce s dokumentem	269
Zpracování řetězců	274
Datum a čas	278
Obrázky	284
Formuláře	288
<i>Zpracování údajů z formulářů</i>	294
<i>Kontrola e-mailové adresy</i>	297
<i>Formátování dat a odeslání formuláře</i>	300
Další možnosti JavaScriptu	303
<i>Detekce prohlížeče</i>	303
<i>Metody objektu LOCATION</i>	304
<i>Metody objektu History</i>	305
CO S HOTOVÝMI STRÁNKAMI	308
<i>Využití jiných zdrojů</i>	308
<i>Umístění stránek na web</i>	309
<i>Umístění odkazů na stránky</i>	311
ZÁVĚREM	313
REJSTŘÍK	316

Správný zdrojový kód a syntaxe jazyka

Nezapomínejte, že vizitkou vaší práce není jen ta část webu, která se zobrazí návštěvníkům stránky na jejich monitorech, ale také zdrojový kód stránek. Validní zdrojový kód, který je napsán podle přijatých standardů, ukazuje na dobrou práci. Webové prohlížeče nejsou primárně konstruovány k tomu, aby luštily, co chtěl autor zdrojovým kódem vlastně říci. Je třeba též nezapomenout na uvedení správné definice verze HTML na začátku každého HTML dokumentu. Tato definice slouží ke snadnějšímu čtení dokumentu prohlížečem a umožňuje správné zobrazení stránek podle přijatých standardů. Validaci stránek lze provést v editoru HTML nebo přímo na internetu. Například na stránkách dříve zmiňovaného konsorcia W3C naleznete on-line validátor nejen pro HTML.



Pamatujte:

- *Internetové stránky by měly být rychlé a přehledné, s dobře členěnou strukturou a intuitivním ovládáním.*
- *Texty zde prezentované pište jednoduše a stručně.*
- *Vytvářejte srozumitelné odkazy a provazujte obsah webu.*
- *Dejte si pozor na množství grafiky a rychlost načítání webových stránek.*
- *Nezapomeňte zkontrolovat validitu zdrojového kódu a funkčnost všech odkazů, než budete stránky publikovat v síti.*
- *Stránky pravidelně aktualizujte.*

Webdesign

Všichni chceme, aby naše stránky vypadaly hezky. Ale HTML není primárně určen pro práci s grafikou. Přesto lze poměrně snadnými kroky kýženého cíle, tedy graficky pěkných stránek, dosáhnout. K tomu se ale váže několik pravidel.

Rozlišení obrazovky

Zcela zásadním parametrem při tvorbě stránek je nutnost brát ohled na různé rozlišení monitorů u uživatelů internetu. V podstatě neexistuje pravidlo, které by stanovilo, pro jaké rozlišení stránky optimalizovat. Proto si zapamatujte toto pravidlo: stránky neoptimalizujte pro žádné rozlišení. Výsledek své práce si prohlédněte jak v menším, tak ve větším rozlišení, abyste se přesvědčili, že jsou stránky stále přehledné a lze se v nich snadno orientovat. Toto pravidlo však poruším trochou statistiky. Většina uživatelů už dnes vzhledem k cenám větších monitorů používá rozlišení kolem 1024x768 pixelů na obrazovku. Je ale stále velké množství těch, kteří ještě používají rozlišení menší, tedy 800x600 pixelů (například uživatelé notebooků). Možná to bude znít tvrdě, ale na uživatele s rozlišením ještě menším prostě zapomeňte.



Pamatujte: Na vaše internetové stránky rozhodně nepatří upozornění typu „Optimalizováno pro rozlišení 1024x768 a vyšší.“ Uživatelé své rozlišení kvůli vašim stránkám měnit nebudou. Proto vytvářejte stránky tak, aby fungovaly správně ve větším i menším rozlišení.

Grafická podoba stránek

Grafické zpracování stránek je rozhodně důležité, a proto je nepodceňujte. Především vždy dobře sladte grafickou podobu stránek. Ne všechny barvy se k sobě totiž hodí. Pěkně nevypadají ani stránky, jež obsahují mnoho různých barev či jejich odstínů.

Webové stránky by měly být kontrastní. To znamená, že text musí být na pozadí čitelný. Ideální kontrast je například černá na bílé, nebo naopak. Jaké barvy použijete, je samozřejmě na vás, ale musíte si uvědomit, že špatně čitelné texty návštěvníci stránek luštit nebudou. Kontrast je též důležitý pro snadnou orientaci na stránkách.

Pokud budete používat grafická tlačítka, mějte ohledy i na uživatele internetu s pomalejším připojením. Proto u tlačítka uvádějte i popisky, které se zobrazí před načtením obrázku. Tak umožníte uživateli nečekat na načtení grafického tlačítka či obrázku.

Pokud jde o celkové množství grafiky, obecně platí, že čím méně, tím lépe pro funkčnost webu. Moderní webdesign však klade důraz na vytříbené grafické zpracování. Všichni přece chceme, aby naše stránky vypadaly hezky. Pokud je to možné, používejte barevná pozadí. A když už se nevyhnete obrázkům, používejte jediné formáty **JPEG** nebo **GIF** (viz následující podkapitola), co nejvíce komprimujte, respektive redukujte počet barev, aby velikost jejich souborů byla únosná. Vždy však mějte na paměti, že kompresí se ztrácí také kvalita obrázku. Proto si stránky prohlédněte i v menším rozlišení, zda je kvalita použité grafiky dostatečná.

V případě, že chcete stránky – lidově řečeno – rozhýbat, uvědomte si nejdříve, že pohyblivé elementy webových stránek odvádějí pozornost návštěvníka od samotného obsahu webu. Na tomto efektu fungují především reklamní bannery. Pokud to je možné, vyhněte se pohyblivým textům, bannerům a podobně. Vaším zájmem je přece hlavně nalákat návštěvníka stránek na jejich obsah.



Pamatujte:

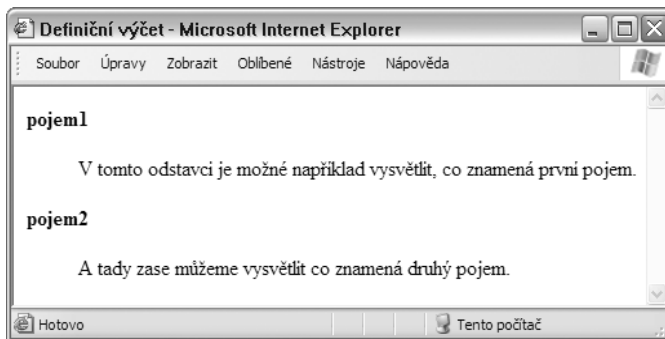
- Dbejte na to, aby se stránky správně zobrazily ve větším i menším rozlišení.
- Myslete na správně sladěnou kombinaci barev.
- Důraz kladte na kontrast a dobrou čitelnost textů.
- U grafických tlačítek a obrázků používejte popisky.
- Obrázky a další grafiku ukládejte ve formátech **JPEG** a **GIF**.
- Nepoužívejte pohyblivé elementy, odvádějí pozornost od obsahu stránek.

```

<DL>
  <DT><b>pojem1</b></DT>
  <DD>
    <p>V tomto odstavci je možné například vysvětlit, co znamená první pojem.</p>
  </DD>
  <DT><b>pojem2</b></DT>
  <DD>
    <p>A tady zase můžeme vysvětlit co znamená druhý pojem.</p>
  </DD>
</DL>

```

Množství kombinací vnořených seznamů či definičních výčtů se samozřejmě meze nekladou. Tento příklad je velmi jednoduchý, takže si sami zkuste doplnit do výčtu další vnořené výčty. Taktéž je možné kombinovat výčty s uspořádanými a neuspořádanými seznamy. Tato možnost pak vytváří ze seznamů poměrně silný nástroj pro formátování větších textových celků a dobře poslouží například i k částečnému zalomení stránky.



Obrázky

V této kapitole si představíme práci s obrázky. Budeme se zabývat především vkládáním obrázků na stránky. Dozvíte se též jak s obrázky na stránce efektivně pracovat. HTML sice není primárně určen pro grafiku, ale vkládání obrázků na web v moderním zpracování webu je nutností, kterou nelze opomenout. Obrázky vkládejte do těla dokumentu pomocí nepárové značky **IMG**, k níž přidělujte příslušné parametry.

```
<IMG src="URL">
```

Tento parametr u tagu **IMG** definuje, jaký konkrétní obrázek budete do stránky vkládat.

```
<IMG alt="popisek obrázku" title="popisek obrázku">
```

Tyto parametry slouží pro vkládání popisku obrázku.

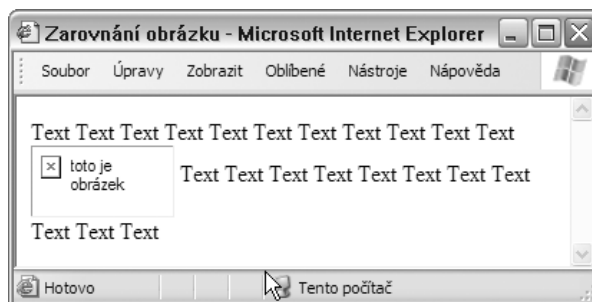
```
<IMG width="px" height="px">
```

Tyto parametry umožňují nastavení pevné výšky (**WIDTH**) a šířky (**HEIGHT**) obrázku.

Aby byl výčet poloh kompletní, podíváme se ještě na jednu ukázkou zarovnání obrázku, a to s polohou **MIDDLE**.

```
<P>
Text Text Text Text Text Text Text Text Text Text Text
<IMG src="obrazek.jpg" alt="toto je obrázek" title="toto je obrázek"
width="100" height="50" align="middle">
Text Text Text Text Text Text Text Text Text Text Text
</P>
```

V této ukázce je dobře vidět, že obrázek se v poloze **MIDDLE** zarovnává svým středem na základnu textu, tedy přímo na řádek.



Na posledním příkladu této kapitoly si ještě předvedeme, jak vypadá rámeček kolem obrázku:

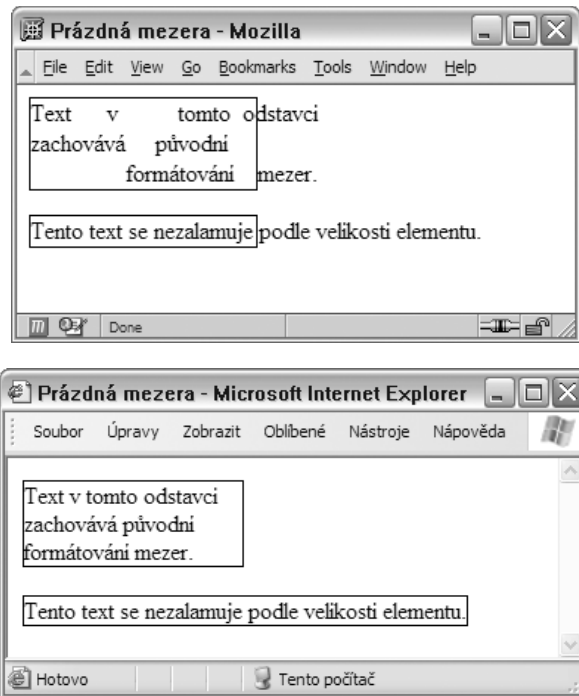
```
<IMG src="obrazek.jpg" alt="toto je obrázek" title="toto je obrázek" width="100"
height="50" border="10">
```

Rámeček kolem obrázku je implicitně černý a jeho barvu lze změnit jediné pomocí kaskádových stylů. V zobrazené ukázce má rámeček velikost 10 pixelů. Taktéž si můžete všimnout, že pokud nezadáte parametr **ALIGN**, který zarovnává obrázek do vybrané polohy vzhledem k dalším elementům HTML na stránce, obrázek se zarovná automaticky doleva.

Odkazy

Odkazy patří mezi nejpoužívanější součásti HTML. Ostatně už sám pojem hypertext naznačuje, že HTML je jazyk provazující text, respektive informace do rozsáhlejších celků. A tím nejrozsáhlejším celkem je internet jako celosvětová síť. Použití odkazů je velmi jednoduché. Můžete jak propojovat jednotlivé texty na vašem webu, tak odkazovat na jiné stránky. Odkazy budou nedílnou součástí vaší stránky. Obejdete se bez nich možná pouze v případě, že vaše internetová prezentace bude mít podobu

Zde vidíte, jaký je rozdíl v zobrazení prázdné mezery v **Mozilla** (správná interpretace) a v **Internet Exploreru** (nesprávná interpretace).



V tomto příkladu byly použity dva odstavce o šířce 150 pixelů se zvýrazněným okrajem. První odstavec předvádí hodnotu **PRE**, druhý hodnotu **NOWRAP**. **Internet Explorer** v tomto případě interpretoval pouze hodnotu **NOWRAP**. Text nezalomil podle šířky odstavce, ale naopak šířku odstavce zvětšil, aby nemusel text zalamovat. Výše uvedená fakta naznačují, jaké problémy s sebou nesou kaskádové styly v některých případech. Proto pečlivě zvažujte, které vlastnosti a hodnoty využít a které ponechat svému osudu.

Rozměry, okraje, rámečky

Tato kapitola se bude zabývat velmi zajímavým a důležitým tématem. Rozměry jako šířku a výšku budete – ať v relativní, nebo absolutní podobě – zadávat u elementů HTML velmi často. CSS nabízí navíc rozšířené možnosti, a tak můžete rozměry zadávat téměř každému elementu. Další část se bude věnovat okrajům. Kaskádové styly umožňují definovat a editovat dva druhy okrajů. Práce s nimi je velmi důležitá pro celkový vzhled stránek, zvláště pokud budete často pracovat s tabulkami. Ale hodí se i pro další elementy HTML. A nakonec diskutované téma rámečků. Rámečky přitom

dobře znáte zvláště z kapitoly o tabulkách v části knihy o HTML. Tato kapitola se tedy dělí na tři podkapitoly, ve kterých se podrobně věnuji nejdříve rozměrům, poté okrajům a nakonec rámečkům. Všechny vlastnosti uvedené v této kapitole jsou nedědičné.

Rozměry

Rozměry jednotlivých elementů HTML, zvláště blokových, jako jsou tabulky, odstavce, nadpisy a tak dále, jsou velmi důležité. Pomocí velikostí jste totiž schopni vytvořit opravdu takovou podobu webu, jakou žádáte. Rozměry, které můžete ovlivňovat, jsou šířka a výška. Jak si jistě pamatujete z části o HTML, i ve zdrojovém kódu lze zadávat zvláště šířku některých elementů. V CSS se k tomu přidává možnost práce s výškou, která byla v HTML téměř u všech elementů znemožněna. Nyní se tedy seznámte se dvěma vlastnostmi – **WIDTH** a **HEIGHT**. Použít je můžete pouze u blokových a nahrazovaných elementů.

WIDTH : délka/%/auto

WIDTH se stará o šířku elementu, ke kterému ji přiřadíte. Na rozdíl od HTML však můžete šířku elementu zadávat mnoha různými způsoby. Samozřejmostí je široká paleta jednotek délky. Šířka zadaná relativně v procentuálním vyjádření se odvozuje z šířky mateřského čili nadřazeného elementu. **AUTO** je implicitní hodnota (šířka je upravena automaticky).

HEIGHT : délka/auto

HEIGHT definuje výšku elementu, u kterého se použije. V tomto případě máte pouze dvě volby jak zapsat hodnoty této vlastnosti. Procenta jsou zde vyloučena. Hodnoty tak mohou nabývat podoby jednotek délky, nebo implicitní hodnoty **AUTO** (výška je upravena automaticky).



Upozornění: V případě, že budete pomocí těchto dvou vlastností měnit velikost obrázku, je nutné nastavit vždy u druhé vlastnosti hodnotu **AUTO**, aby byl zachován poměr stran. V případě, že obě vlastnosti, jak **WIDTH**, tak **HEIGHT**, budou nabývat hodnoty **AUTO**, velikost bude nastavena tak, aby vyhovovala skutečným rozměrům daného elementu.

Styly:

```
TABLE { width : 300px; height : 250px; text-align : center }  
IMG.maly { height : 20px }
```

Zdrojový kód:

```
<TABLE border="1">  
<TR>  
<TD><IMG src="c-media_logo.gif"></TD>
```

(první za otazníkem). V případě, že podmínka splněna nebude, bude vykonán výraz za dvojtečkou, jak naznačuje ukázka výše (příklad ternárního operátoru).

Proměnné

Proměnná je jedním ze základních prvků každého programovacího jazyka. Je to místo v paměti, jemuž je přiřazen název. Proměnná v sobě dokáže uchovávat hodnoty datového typu. S proměnnou lze samozřejmě dále manipulovat (například provádět operace). Přitom v jednom čase má pouze jednu hodnotu. Novou proměnnou je třeba nejdříve deklarovat, k čemuž slouží klíčové slovo **VAR**.

```
var proměnná // základní deklarace proměnné  
nebo
```

```
var proměnná = výraz // deklarace proměnné s úvodním přiřazením hodnoty
```

Každé základní deklaraci proměnné by podle normy mělo předcházet klíčové slovo **VAR**. Přitom proměnné při základní deklaraci nemusíte přiřazovat hodnotu. Následující příklad naznačuje jak proměnnou deklarovat v praxi:

```
var cislo = 1;  
var text = 'Text jako výraz';  
var logicka = true
```

V tomto případě jsme nejdříve definovali proměnnou **cislo** a přiřadili ji číselnou hodnotu **1**. V druhém případě byla vytvořena proměnná **text**, jejíž hodnotou je textový řetězec. Třetí příklad naznačuje jak vytvořit proměnnou s logickou hodnotou **TRUE**.



Pamatujte: JavaScript rozlišuje malá a velká písmena! Relativně stejný název tak můžete napsat mnoha způsoby a JavaScript jej bude chápat jako více názvů pro různé entity (například proměnná **cislo** je jiná proměnná než **Cislo**).

Jméno proměnné se může skládat z libovolné kombinace čísel a písmen a znaku podtržítka (**_**). Prvním znakem však nemůže být nikdy číslo. Proměnnou je taktéž možné deklarovat buď jako globální, nebo jako lokální. A jaký je mezi nimi rozdíl? **Globální proměnná** je definována v obecné části skriptu (mimo prvky JavaScriptu). Ve skriptu se však může vyskytovat ještě další proměnná se stejným názvem – například uvnitř funkce. Uvnitř této funkce pak bude skript pracovat s touto **lokální proměnnou**.



Poznámka: Deklarovat proměnnou před jejím použitím pomocí klíčového slova **VAR** není vždy nezbytné. Nicméně stalo se to programátorským zvykem. První použití proměnné znamená totiž zároveň i její deklaraci. Klíčové slovo **VAR** je však nutné uvádět v případě, že deklarujete uvnitř funkce lokální proměnnou, jež má shodný název s proměnnou globální (ta je umístěna mimo tuto funkci). Pokud byste v takovém případě nepoužili deklaraci slovem **VAR**, prohlížeč by interpretoval globální proměnnou.

Proměnné v JavaScriptu mohou nabývat hodnoty jakéhokoli datového typu. Je tedy možné například přiřadit proměnné textový řetězec a později stejné proměnné zase číslo. Důvodem je fakt, že JavaScript je netyповý jazyk. Deklarace proměnných tedy nemusí specifikovat datový typ proměnné. Výsledkem této skutečnosti je i další vlastnost tohoto jazyka – hodnoty proměnné jsou vždy vhodně a automaticky konvertovány. Více naznačuje tato ukázka:

```
var i = 10;  
i = i + 'deset'
```

Proměnná **i** bude mít po zpracování tohoto skriptu hodnotu **10deset**, jež bude řetězcem. A co se vlastně stalo? Nejdříve jsme úvodní deklarací přiřadili proměnné **i** číslo **10**. Poté byl k této hodnotě přidán textový řetězec **deset**. JavaScript převedl původní číselnou hodnotu na řetězec a ten sečetl s druhým řetězcem. Výsledkem této operace je opět textový řetězec.

Příkazy

Vzpomeňte si nyní na definici výrazu JavaScriptu, uvedenou v příslušné podkapitole. Tyto výrazy však ve své podstatě ještě nic nedělají, slouží pouze k vyhodnocení. Aby se něco opravdu stalo, je potřeba použít příkaz. Ten je podobný povelu. V podstatě celý skript je souborem těchto příkazů, jež oddělujeme středníkem (;).

Mezi nejjednodušší příkazy patří **výrazy s vedlejším účinkem**, například inkrementace či dekrementace proměnné (mění hodnotu proměnné). Jednoduchým příkazem je ale i volání funkce (funkcím je věnována jedna z následujících podkapitol). Existují samozřejmě i příkazy složitější. Například složené příkazy, jež uzavíráme do složených závorek (**{** a **}**), se skládají z několika příkazů oddělených středníkem. Tyto řádky kódu pak působí jako jeden příkaz a používají se tam, kde JavaScript očekává pouze jeden příkaz. Nejčastěji se ale setkáte s příkazy v rámci cyklů či podmínek (například klíčová slova **IF**, **WHILE** či **FOR**), jimž se věnují dvě následující podkapitoly. Příkazem je však i klíčové slovo, jímž je deklarována proměnná (**VAR**), nebo definována funkce (**FUNCTION**).

Další možnosti JavaScriptu

V závěrečné kapitole části knihy o JavaScriptu se ještě zmíním o některých dalších možnostech tohoto skriptovacího jazyka, na něž nezbylo místo v jiných kapitolách. Nejprve se podíváme, jak je možné detekovat prohlížeč použitý k zobrazení vašich stránek. Taktéž se zmíním o některých dalších možnostech objektu **LOCATION** a představím některé metody objektu **HISTORY**.

Detekce prohlížeče

Detekovat prohlížeč, který používá návštěvník vašich stránek, se může zdát jako příjemné zpestření. Využít detekci ale můžete zvláště tam, kde chcete skript optimalizovat pro každý prohlížeč zvlášť. K detekci prohlížeče slouží objekt **NAVIGATOR** a díky několika vlastnostem zjistíte řadu zajímavých věcí.

appName

Název prohlížeče zjistíte pomocí této vlastnosti objektu **NAVIGATOR**.

appVersion

Tato vlastnost vrací verzi použitého prohlížeče.

browserLanguage, language

Tyto dvě vlastnosti zjišťují jazykové nastavení použitého prohlížeče. S vlastností **BROWSERLANGUAGE** umí pracovat **Internet Explorer**, zatímco s **LANGUAGE** **Mozilla**. **Opera** si poradí s oběma vlastnostmi.

platform

Vlastnost **PLATFORM** slouží ke zjištění typu operačního systému nebo také platformy, na které prohlížeč funguje.

Tyto vlastnosti pak shrnuje následující ukázka. Půjde pouze o skript, proto nebudu uvádět tělo HTML dokumentu, jež je v tomto případě prázdné.

```
document.write('Název prohlížeče: '+ navigator.appName + '<BR>');  
document.write('Verze prohlížeče: '+ navigator.appVersion + '<BR>');  
document.write('Jazyk prohlížeče: '+ navigator.browserLanguage + '<BR>');  
document.write('Platforma prohlížeče: '+ navigator.platform + '<BR>')
```

V tomto příkladu byl použit krátký skript, jenž popisuje použitý prohlížeč. V tomto případě jde o **Internet Explorer** verze **6.0**. Pro zjištění platformy prohlížeče