Překročte hranice Webu 2.0

Den Odell

# JavaScript

## Průvodce programováním ajaxových aplikací

Tvorba interaktivních RIA aplikací

Ladění výkonu aplikace pro různé prohlížeče

Začlenění interaktivní grafiky a multimédií do projektu

WWW

Ke stažení zdrojové kódy příkladů z knihy

PRESS Apress®

Den Odell

# JavaScript
## Průvodce programováním ajaxových aplikací

Computer Press, a. s.
Brno
2010

# JavaScript
## Průvodce programováním ajaxových aplikací

**Den Odell**

**Computer Press, a. s.**, 2010. Vydání první.

**Překlad:** Ondřej Baše
**Odborná korektura:** Lukáš Krejčí
**Jazyková korektura:** Vladimír Šenkýř
**Vnitřní úprava:** Petr Klíma
**Sazba:** Petr Klíma
**Rejstřík:** Daniel Štreit
**Obálka:** Martin Sodomka

**Komentář na zadní straně obálky:** Martin Herodek
**Technická spolupráce:** Jiří Matoušek, Zuzana Šindlerová, Dagmar Hajdajová
**Odpovědný redaktor:** Martin Herodek
**Technický redaktor:** Jiří Matoušek
**Produkce:** Petr Baláš



Autorizovaný překlad z originálního anglického vydání Pro JavaScript™ RIA Techniques: Best Practices, Performance, and Presentation.


**Computer Press, a. s.**,
Holandská 8, 639 00 Brno

Objednávky knih:
http://knihy.cpress.cz
distribuce@cpress.cz
tel.: 800 555 513

ISBN 978-80-251-2733-9
Prodejní kód: K1743
Vydalo nakladatelství Computer Press, a. s., jako svou 3567. publikaci.

# Obsah

KAPITOLA 10



KAPITOLA 11



KAPITOLA 12

# Úvod

RIA (Rich Internet Application) aplikace, neboli webové aplikace, jsou takové weby, které se snaží smazat rozdíly mezi webovým prohlížečem a klasickými desktopovými aplikacemi. Správa vaší poštovní schránky prostřednictvím webových stránek typu Google Gmail, Yahoo! Mail nebo Microsoft Windows Live Hotmail je v každém ohledu stejně intuitivní jako pomocí poštovních klientů Microsoft Outlook nebo Apple Mail. Obnovení webové stránky při provádění akcí je nežádoucí a spíše očekáváte, že když poštovní server přijme novou zprávu, objeví se bezprostředně v doručené poště. Webové stránky, které se chovají tímto způsobem, jakoby vybočují z klasického modelu, kde je obvyklé, že klepnete na nějaký odkaz nebo odešlete formulář, abyste například aktualizovali obsah online diskusního fóra. Tento rozdíl vedl k tomu, že někteří lidé začali RIA aplikace označovat termínem Web 2.0, což může vést k představě, že vznikla nová verze celosvětové sítě.

V některých ohledech skutečnosti došlo k inovaci, ale nikoliv k inovaci Webu jako takového. Zlepšily se především webové prohlížeče, kterými prohlížíte své oblíbené webové stránky. V průběhu posledních několika let se postupně rozrůstala nabídka funkcí hlavních webových prohlížečů. Kromě toho se začala objevovat snaha výrobců webových prohlížečů o sjednocení webových technologií, díky současným možnostem jazyka JavaScript a objektovému modelu dokumentů DOM lze aktualizovat webovou stránku pomocí dat, která se dynamicky načtou z webového serveru. Web už tedy není jen prostorem statických stránek.

Tuto knihu jsem napsal zejména proto, abych vám pomohl pochopit skutečnou sílu jazyka JavaScript, s jehož pomocí můžete přidávat ke svým stránkám dynamické prvky a vytvářet opravdové RIA aplikace. (Předpokládám, že již máte nějaké znalosti s jazyky HTML, CSS a JavaScript.) S velkými možnostmi však přichází i velká zodpovědnost. Zejména budu klást důraz na to, abyste pochopili, jak je důležité vytvářet design, který má vaše návštěvníky překvapit, nikoliv odradit. Rovněž je důležité, abyste uměli vytvářet aplikace, které se chovají a vypadají lépe než běžná statická webová stránka. Uvidíte, jak vytvářet vlastní prvky uživatelského rozhraní, které nezničí použitelnost a přístupnost.

Na konci této knihy byste měli mít dostatek odvahy k tvorbě vlastního webu nebo RIA aplikace, a to s vědomím, že vytváříte robustní, spolehlivou, efektivní, pěknou a vysoce přístupnou aplikaci.

# ČÁST I

# Praxí osvědčené postupy

V této první části knihy vám představím několik osvědčených postupů tvorby aplikací RIA (Rich Internet Application). Pokud se budete těmito postupy řídit, budete schopní vytvářet základní strukturu webových stránek, která je škálovatelná od jediné stránky s několika řádky kódu až po tisíce stránek obsahujících tisíce řádků kódu. Ukážu vám, jak tyto postupy aplikovat, aniž by se údržba aplikace nebo oprava chyb stala frustrující činností, a to jak v průběhu vývoje, tak i v budoucnu.

# KAPITOLA 1

# Vybudování pevného základu

**V této kapitole:**

- Přehled praxí osvědčených postupů
- Obecné praxí osvědčené postupy
- Osvědčené postupy pro značkování: sémantické HTML
- Praxí osvědčené postupy pro formátování: CSS
- Uspořádání složek, souborů a prostředků
- Nastavení vývojového prostředí

Jestliže čtete tuto knihu, je celkem pravděpodobné, že jste již někdy zažili ten skvělý pyšný pocit, který se dostaví, když vytvoříte a představíte svou novou webovou stránku. Možná jste dokončili takový projekt sami, možná jako součást týmu. Možná se jednalo o jednoduchý web s několika stránkami, které prezentovaly vaši osobnost na Internetu pro vaše nejbližší přátele, nebo šlo o vybroušenou RIA aplikaci s funkcemi sociální sítě cílené milionům uživatelů. V každém případě gratuluji k dokončení projektu, zasloužíte si být na sebe pyšní.

Když se ohlédnete k počátkům vaše projektu se zkušeností, kterou jste získali v průběhu jeho vývoje, vsadím se, že vás napadne alespoň jedna věc, kterou jste mohli udělat jinak a ušetřit si tak zbytečné mlácení hlavou o zeď. Pokud začínáte s vývojem webových aplikací, možná tou chybou bylo, že jste zapomněli udělat zálohu starší verze zdrojových kódů a museli jste se značnou dobu snažit obnovit změny po neočekávaném výpadku proudu. Možná si přejete, abyste se nerozhodli pro softwarovou knihovnu třetí strany, která se zpočátku projektu zdála velmi vhodnou, ale postupně se z ní vyklubala obrovská ztráta času a úsilí. V průběhu mé kariéry jsem se dostal přesně do těchto situací a vyšel jsem z nich o něco moudřejší. Poučil jsem se z chyb a snažil se nové znalosti aplikovat v dalších projektech.

Na základě své zkušenosti a také toho, co jsem se naučil od ostatních , jsem si vytvořil efektivní a smysluplný přístup k vývoji webových aplikací. Když smícháte tento přístup a chytré techniky, měli byste minimalizovat počet těchto nepříjemných situací a zajistit, aby vývoj vaší další webové aplikace probíhal hladčeji.

# Přehled praxí osvědčených postupů

Začněme tím, co je myšleno pod termínem *praxí osvědčený postup*. Pokud již nějakou dobu programujete, určitě jste se s tímto výrazem setkali jako s označením určité programovací techniky nebo přístupu. Je to poměrně často používaný termín, ale měl by se používat s rozmyslem. Vysvětlím proč.

## Proč tyto postupy označujeme jako „osvědčené“?

Vývoj webových aplikací se neustále mění. Zvyšuje a snižuje se popularita jednotlivých webových prohlížečů, ne vždy od sebe přejímají dobré funkce a technologie používané pro tvorbu webových stránek zobrazovaných v těchto prohlížečích jsou stále poměrně nedokonalé. Neustále vznikají jejich další revize a aktualizace. V takovém nestálém prostředí může být řešení, které nyní považujeme za osvědčené, za půl roku zastaralé.

Použití slova *osvědčené* naznačuje, že jste se rozhodovali na základě nějakých srovnávacích testů nebo vědeckého výzkumu. Ve skutečnosti však k takovým testům dochází vzácně. Proto byste měli opatrně popřemýšlet nad technikami, technologiemi a komponentami, které byly označeny jako *osvědčené praxí*. Zamyslete se a uvažte, zda splňují vaše požadavky jako programátora, požadavky koncového uživatele a pokud je to nutné i požadavky zákazníka, pro kterého webovou aplikaci vytváříte.

Pokyny, pravidla a techniky uvedené v této kapitole jsem osobně vyzkoušel a můžu potvrdit, že jsou vhodné pro vývoj skutečných webových aplikací. Považuji je za to nejlepší, co lze v současné době použít. Samozřejmě, že některé z nich mohou být zastaralé v době, kdy čtete tuto knihu, a proto vám doporučuji, abyste změny v tomto odvětví bedlivě sledovali. Čtěte časopisy, zaregistrujte se do dis-

kuzních fór, diskutujte s ostatními programátory a hledejte informace na Internetu. Rozsáhlý seznam zdrojů najdete na mých osobních webových stránkách na adrese *http://www.denodell.com/*, které můžeme využít jako výchozí bod pro své hledání.

Pokud budete držet krok se změnami těchto praxí osvědčených postupů, měli byste se udržet v popředí odvětví vývoje webových aplikací vyzbrojeni sadou nástrojů a technik, které mohou učinit vaší každodenní práci efektivnější a konstruktivnější.

Nebojte se při tvorbě svých webových stránek vracet zpět a přepisovat zdrojový kód. Nikdo nenapsal webovou stránku od začátku do konce, aniž by musel měnit zdrojový kód. Nevěřte ani chvilku tomu, že příklady zdrojového kódu v této knize, nebo v jakékoliv jiné knize, byly napsány perfektně hned napoprvé. Tato rada a vaše vlastní zkušenosti vám velmi ulehčí práci, proto využijte každé příležitosti k tomu, abyste se stali nejlepším webovým vývojářem, jakým můžete být.

## Kdo těží z praxí osvědčených postupů?

Pravda je, že každý může využít praxí osvědčené postupy ve svém zdrojovém kódu. Prohlédněte si následující seznam a použijte tato kritéria při rozhodování o vhodnosti pokynů, technik a technologií pro vaše webové stránky.

#### Weboví vývojáři

Praxí osvědčené postupy začínají doma. Struktura stránek, která vyhovuje vám a vašim kolegům, vám velmi usnadní život a sníží počet potenciálních problémů, které mohou vzniknout špatným programováním.

- Bude můj zdrojový kód splňovat doporučení konsorcia W3C (World Wide Web Consortium)?
- Budou mé webové stránky použitelné, i když nebude příslušná technologie nebo zásuvný modul dostupný?
- Projde můj zdrojový kód testováním a validací?
- Je můj zdrojový kód srozumitelný, dobře strukturovaný a udržovatelný?
- Lze zvláštní stránky, části a prostředky přidat k webovým stránkám bez výrazného nadbytečného úsilí?
- Lze moje webové stránky lokalizovat do různých světových jazyků bez příliš velkého úsilí?

#### Vyhledávací roboti a další automatizované systémy

Ať už tomu věříte, nebo ne, velké procento síťového provozu pochází od automatizovaných systémů a skriptů, například od vyhledávacích robotů, snímačů obrazovek a nástrojů pro analýzu sítě. Navrhování stránek pro tyto roboty je stejně důležité jako pro ostatní skupiny uživatelů.

- Objeví se mé stránky správně ve výsledcích vyhledávání vyhledávacího robota, když uživatel zadá vhodné výrazy?
- Může robot nebo skript snadno přečíst nebo rozebrat můj zdrojový kód, ať už tak činí z jakýchkoliv důvodů?

#### Koncoví uživatelé

Nejdůležitější uživatelé vašich webových stránek jsou vaši návštěvníci, proto úlohou s nejvyšší prioritou je zaručit, aby váš kód pro ně skutečně fungoval.

- Budou mé webové stránky použitelné a přístupné pro libovolný webový prohlížeč nebo zařízení bez ohledu na jeho stáří, velikost obrazovky nebo metodu vstupu dat?
- Kdyby byl obsah mých webových stránek předčítán nahlas čtečkou obrazovky, dával by posluchači nějaký smysl?
- Mohu se spolehnout na to, že mé webové stránky se nebudou chovat chybně nebo nezobrazí chybovou zprávu, pokud budou použity způsobem, se kterým jsem nepočítal?
- Je možné mé webové stránky vyhledat vyhledávacím robotem nebo jiným nástrojem, pokud uživatel zadá vhodné hledané výrazy?
- Mohou uživatelé snadno použít lokalizovanou verzi webových stránek, pokud je nějaká k dispozici?

# Obecné praxí osvědčené postupy

Pokud jste jako většina vývojářů, pravděpodobně bude chtít nejvíce času trávit tvorbou prvků atraktivního uživatelského rozhraní a skvěle vypadajících webových stránek, a ne přeprogramováním jádra zdrojového kódu, které bylo špatně architektonicky navrženo. Je velmi důležité, aby byl zdrojový kód dobře udržovatelný. Bez smysluplné struktury a čitelnosti bude jeho údržba stále těžší a těžší. Vezměte prosím na vědomí, že všechny pokyny v této kapitole se vám, jako programátorovi, snaží co nejvíce ulehčit práci.

## Definujte cíle projektu

Při programování webové stánky byste měli zvážit dvě důležité věci:

- Jak ji bude koncový uživatel používat?
- Jak ji mohou další vývojáři měnit?

Nezapomeňte, že koncovým uživatelem nemusí být člověk. Kdybyste někdy prohlíželi protokol požadavků serveru, objevili byste, že spousta návštěvníků vašich webových stránek jsou ve skutečnosti vyhledávací roboti, RSS čtečky a další online služby schopné přečít holý obsah webových stránek a změnit jej na něco jiného.

Tento automatizovaný přístup se pravděpodobně v několika příštích letech více rozšíří, jak se například RSS vlákna stanou běžnějšími. Například obsah populární encyklopedie Wikipedia (*http://www.wikipedia.org/*) se již používá na různých místech celosvětové pavučiny, například v aplikaci Google Maps (*http://maps.google.com/*), kde jsou články z encyklopedie rozmístěny podle geografické pozice popsané v obsahu článku.

Yahoo! a další společnosti provozující vyhledávače nějakou dobu tlačily na vývojáře, aby do svých stránek vkládali značky popisující obsah, díky nimž mohou vyhledávací roboti lépe pochopit obsah stránek a také prezentovat své výsledky jiným způsobem. Například recepty by mohly být zobrazeny s obrázky a ingrediencemi; výsledky týkající se filmů by mohly obsahovat recenze a seznam kin ve vaší blízkosti, v nichž se daný film právě vysílá. Možnosti jak propojit váš zdrojový kód se zdrojovými kódy jiných vývojářů, jsou obrovské. Označením svého obsahu správným způsobem zajistí-

te, že celý systém bude srozumitelný, logický a souvislý, což pomůže uživatelům rychleji najít informace, které hledají.

Abyste zajistili, že ostatní vývojáři (včetně vás, pokud se k projektu vrátíte po delší pauze) porozumí vašemu zdrojovému kódu, musíte zvážit, jaké obvyklé úlohy související s údržbou vašich stránek budete provádět. Tyto úlohy běžně spadají do následujících čtyř kategorií:

- Provádění úprav stávajících stránek.
- Přidání nových stránek.
- Opětovný návrh nebo úprava rozvržení stránky.
- Podpora pro koncové uživatele, kteří potřebují stránku v dalších jazykových či regionálních verzích.

Tím, že budete přemýšlet o budoucích úlohách předem, si usnadníte práci, protože nebudete muset tolik přepisovat zdrojový kód a rozdělovat soubory. Vítej zpět, zdravý rozume.

## Mějte na paměti základní pravidla

Jak si můžete být jisti, že děláte věci správně? Následujících sedm pravidel to stručně shrnuje:

- Vždy se řiďte vyspělými a otevřenými webovými standardy s kvalitní podporou.
- Mějte na paměti rozdíly mezi implementacemi jazyků HTML, CSS a JavaScript v jednotlivých webových prohlížečích a naučte se, jak se s nimi vypořádat.
- S podporou jazyka HTML počítejte, ale vytvořte webové stránky, které budou použitelné i přes absenci dalších technologií webového prohlížeče – například jazyka CSS, JavaScript nebo zásuvných modulů.
- Složky a soubory pojmenovávejte konzistentně a seskupujte soubory podle jejich účelu, struktury stránek a jazyka.
- Pravidelně pročišťujte nadbytečný zdrojový kód, soubory a složky, čímž získáte čisté a úhledné jádro zdrojového kódu.
- Pište efektivní zdrojový kód.
- Zbytečně nepoužívejte technologie bez zjevného důvodu.

Nyní si projděme každé základní pravidlo zvlášť.

#### Řiďte se vyspělými a otevřenými webovými standardy s kvalitní podporou

Na počátku 90. let 20. století vynalezl chytrý muž Tim Berners-Lee, který pracoval ve výzkumné organizaci CERN (evropská organice pro jaderný výzkum, *http://www.cern.ch/*), něco, co dnes známe jako celosvětovou pavučinu (World Wide Web, zkráceně web). Přišel s návrhem domovských stránek, značkovacího jazyka HTML a propojených hypertextových odkazů, které tvoří základ celosvětové pavučiny. Rovněž vytvořil první webový prohlížeč, aby mohl předvést svůj vynález.

Jeho projekt se poměrně rozrostl a spotřeboval spoustu prostředků organizaci CERN. Když se v organizaci CERN rozhodli, že své prostředky a talent raději dají na projekt urychlovače částic, Tim Berners-Lee si založil novou organizaci, která by mohla pokračovat v tvorbě standardů jazyka HTML a souvisejících technologií. Tato nová organizace, konsorcium W3C (World Wide Web Consortium, *http://www.w3.org/*), vznikla v říjnu roku 1994.

Od svého vzniku vydalo konsorcium W3C přibližně 110 standardů a praktických postupů vztahujících se k celosvětové pavučině. Pro čtenáře této knihy jsou nejužitečnější standardy náležící jazyku HTML (a XHTML), jazyku CSS a objektovému modelu dokumentů DOM spolu s jazykem JavaScript (rovněž bývá označován jako ECMAScript, protože JavaScript je obchodní značka společnosti Sun Microsystems).

V době vzniku konsorcia W3C se také objevily první webové prohlížeče – Netscape Navigator v prosinci roku 1994 a Internet Explorer (IE) od společnosti Microsoft v srpnu roku 1995. Oba webové prohlížeče měly podobný základ zdrojového kódu a zobrazovaly webové stránky stejným způsobem. Samozřejmě webové stránky té doby se vizuálně velmi lišily od těch současných, a proto nebylo těžké tohoto souladu dosáhnout.

Když přejdeme k roku 1996, situace začíná být zajímavější. Společnost Microsoft zavedla základní podporu nového doporučení konsorcia W3C do prohlížeče IE 3, a to doporučení CSS level 1. Toto doporučení definovalo způsob, jak nastavit písmo, barvu, zarovnání textu, okraj a vnější okraj většině elementů stránky. Společnost Netscape se rychle přizpůsobila a soutěživost mezi těmito dvěma výrobci začínala nabírat na intenzitě. Obě společnosti se snažily implementovat nová a budoucí doporučení konsorcia W3C, a to často ještě předtím, než byly vydány finální verze těchto doporučení.

Tato rozdílná podpora standardů ve webových prohlížečích přirozeně vedla ke zmatení webových vývojářů, kteří obvykle navrhovali stránky pro jediný prohlížeč. Z tohoto důvodu se koncoví uživatelé setkávali na webových stránkách se zprávami typu „Tato webová stránka funguje jen v prohlížeči Internet Explorer. Prosím změňte svůj webový prohlížeč."

Samozřejmě doporučení konsorcia W3C jsou pouze doporučeními pro výrobce webových prohlížečů a pro vývojáře. Jako vývojáři je můžete považovat za užitečné jen do té míry, do které jsou implementované v oblíbených webových prohlížečích. Postupně se samozřejmě výrobci webových prohlížečů snaží směřovat k tomu, aby jejich implementace odpovídaly aktuálním webovým standardům. Bohužel však starší verze webových prohlížečů se slabou podporů webových standardů stále existují a lidé je pořád používají, proto s nimi musí weboví vývojáři počítat.

Z toho plyne, že byste měli sledovat aktuální podporu standardů v prohlížečích daleko více než poslední doporučení konsorcia W3C. Pokud se standard rozšíří, měli byste jej začít používat. V opačném případě je lepší se mu vyhnout.

#### Řešení problémů s rozdíly mezi prohlížeči

Webové prohlížeče jsou pravidelně aktualizovány a často přináší lepší podporu stávajících doporučení konsorcia W3C a někdy se pokouší implementovat budoucí doporučení.

V minulosti se webové prohlížeče velmi lišily v implementaci existujících doporučení. To se týká i podpory novějších doporučení v prohlížečích. To znamená, že vývojáři musí sledovat změny v prohlížečích a znát omezení jednotlivých prohlížečů.

Na druhou stranu většina uživatelů webových prohlížečů jen zřídka své webové prohlížeče aktualizuje tak, jak by si vývojáři přáli. Dokonce i prohlížeče, které mají zapnuté automatické aktualizace, se před stažením poslední verze nejprve zeptají uživatele. Většinu uživatelů tato oznámení ruší, protože chtějí co nejrychleji spustit webový prohlížeč, a proto aktualizaci odloží.

Jako vývojáři musíte vědět, že existují různé webové prohlížeče a jejich verze (celkově přibližně 10 000 různých verzí a stále jich přibývá). Nemůžete tedy předpokládat, jaký webový prohlížeč bude koncový uživatel používat k prohlížení vašich stránek.

To, co můžete zjistit ze statistik, například Net Applications' Market Share (*http://marketshare.hitslink.com/*), je, že mezi pět nejoblíbenějších webových prohlížečů patří IE od společnosti Microsoft, Firefox od společnosti Mozilla, Safari od společnosti Apple, Opera od společnosti Opera Software a Chrome od společnosti Google. Pomocí těchto pěti prohlížečů prohlíží webové stránky celkově 99% uživatelů. Pokud se však budete spoléhat jen na tyto statistiky, ochudíte se o rychle vzkvétající trh s mobilními zařízeními, a proto se vyplatí aktuální vývoj na trhu s webovými prohlížeči bedlivě sledovat.

Testováním vašich stránek v různých prohlížečích a operačních systémech odhalíte ty části zdrojového kódu, které způsobují, že je rozdílné webové prohlížeče interpretují jinak. Minimalizace počtu těchto rozdílů je jedním s nejtěžších úkolů webového vývojáře a tímto se povolání webového vývojáře značně odlišuje od ostatních povolání souvisejících s vývojem softwaru. S tímto problémem se musí počítat od začátku vývoje nového projektu, jen tak se vyhnete šíleným půlnočním programovacím seancím a nedodržení termínu.

Nejlepším přístupem je vytvořit zdrojový kód v jazycích HTML, CSS a JavaScript, který vytvoří základní šablonu stránek předtím, než začnete psát kód specifický pro každou stránku. Potom otestujte tuto páteřní strukturu v co největším počtu webových prohlížečů, operačních systémů a rozlišeních obrazovky a okna. Odlaďte zdrojový kód tak, aby se šablona zobrazovala korektně, a potom můžete začít vkládat specifický kód a obsah.

Určitým zdrojem rozdílů bývá rozdílná interpretace barev napříč webovými prohlížeči. Některé webové prohlížeče načítají informace o barevném profilu z obrázkových souborů, jiné to nepodporují. V důsledku toho se některé obrázky a barvy mohou zobrazit v různých prohlížečích odlišně, a proto se vyplatí zkontrolovat, zda v návrhu vašich webových stránek nedochází ke špatné interpretaci barev.

Jednotlivé části stránky byste měli vytvářet samostatně a testovat je v co možná největším počtu webových prohlížečů. Čím dříve začnete testovat, tím méně chyb budete muset později opravovat. K závěru projektu běžně programátoři pociťují tlak požadavků svých zákazníku na změny na poslední chvíli, proto je vhodné mít do této fáze opravených co nejvíce chyb.

#### Počítejte jen s podporou jazyka HTML

Vaše webové stránky musí být viditelné a funkční v libovolném prohlížeči a zařízení, přitom se však nesmí spoléhat na jazyk CSS, JavaScript, ani na zásuvné moduly. Přestože jazyk CSS, JavaScript a zásuvné moduly můžou přidávat další obsah nebo nové rozvržení a funkce, koncoví uživatelé by měli mít přístup k podobnému obsahu a funkcím i bez těchto technologií. Pokud například pomocí technologie Flash vytvoříte animovanou nabídku, musíte zajistit, že bude k dispozici podobná nabídka i v jazyku HTML, jinak přijdete o podstatnou část uživatelů.

To samozřejmě velmi ovlivní způsob, jak budete vyvíjet webové stránky. Začnete od základů v jazyku HTML a budete hlídat, jestli neztratíte důležitou funkci, pokud v prohlížeči nejsou povoleny nebo zcela schází určité funkce. Každá „vrstva" zdrojového kódu by měla být čistá, neměla by tedy obsa-

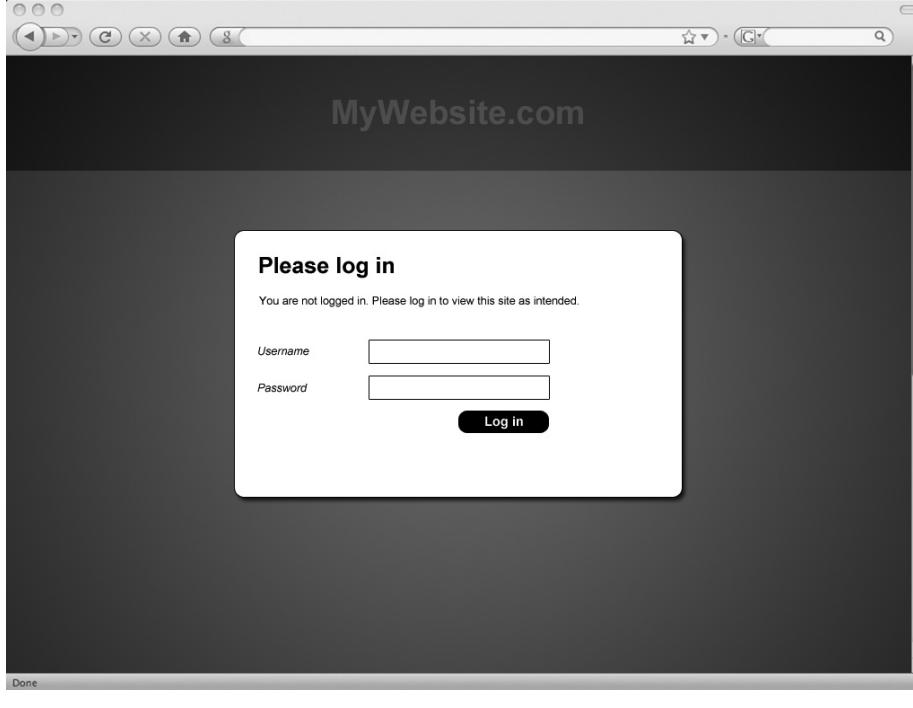
hovat žádná pravidla jazyka CSS nebo zdrojový kód jazyka JavaScript. Tyto prvky by měly být umístěny v samostatných souborech.

V moderních webových aplikacích se často setkáte s tím, že prohlížeč komunikuje se serverem prostřednictvím jazyka JavaScript, což znamená, že tyto komunikační body musí existovat, i když vypnete JavaScript v prohlížeči. Moderní webové aplikace například umožňují odeslat data formuláře webovému serveru bez nutnosti obnovit webovou stránku. V takovém případě musíte zajistit, aby data dorazila k webovému serveru i bez přítomností funkcí jazyka JavaScript. Nakládejte tedy s jazykem JavaScript jako s příjemným oživením, nikoliv jako s nezbytnou součástí.

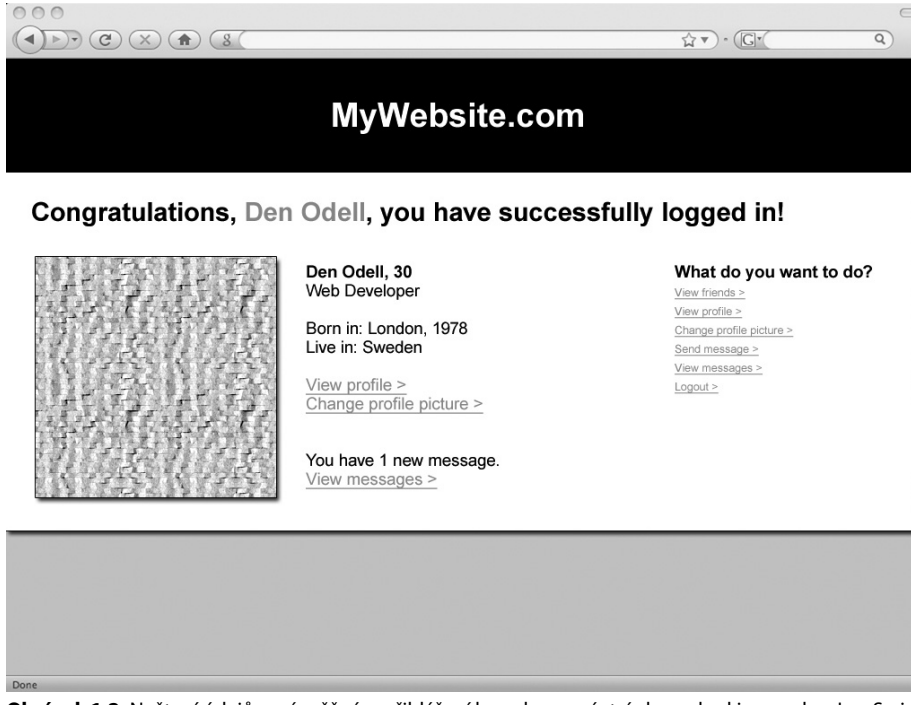
Tento princip bývá někdy označován termínem *progresivní rozšíření*, což odpovídá tomu, že k HTML kódu přidáváte navíc speciální funkce, nebo také *elegantní sestup*, což vyplývá ze skutečnosti, že po odstranění funkcí vždy obdržíte funkční webovou stránku. Toto pravidlo je základním principem *přístupnosti*, která se snaží webové stránky zpřístupnit všem, bez ohledu na použitý prohlížeč nebo zařízení.

Dané pravidlo nejlépe pochopíte na příkladech z běžného života, proto vám nyní dva takové předvedu.

Předpokládejme, že vaše webová aplikace má tlačítko, které po kliknutí zobrazí modální dialogové okno, které vidíte na obrázku 1.1. Jakmile uživatel vyplní formulář a odešle jej příslušným tlačítkem, JavaScript pošle přihlašovací údaje webovému serveru a následně aktualizuje pouze některé elementy stránky, namísto celé stránky, a to na základě výsledku procesu přihlášení uživatele, jak vidíte na obrázku 1.2.




**Obrázek 1.1.** Modální dialogové okno pro přihlášení.



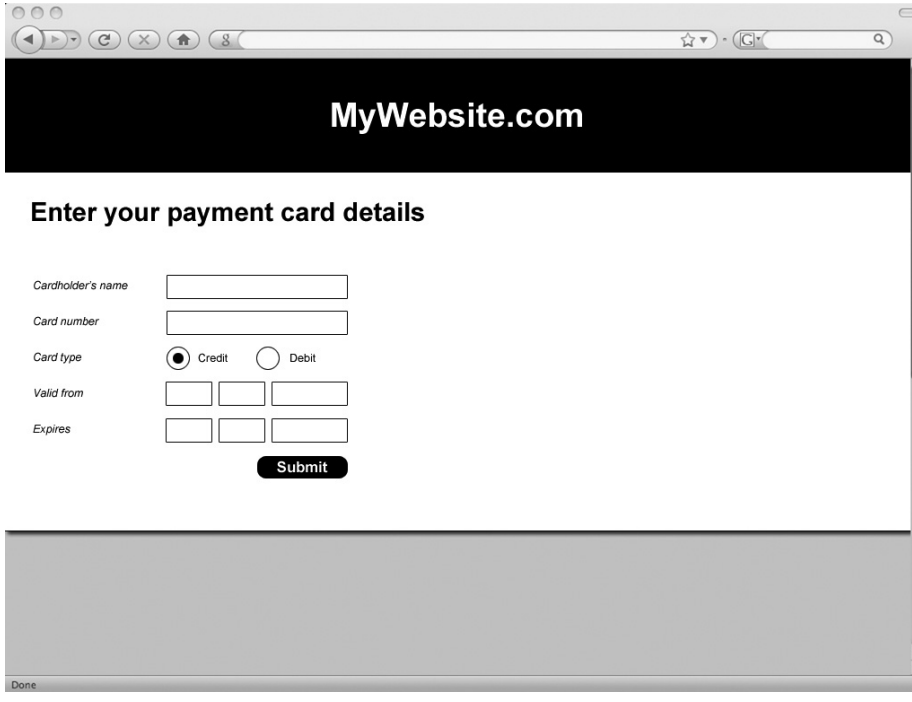

**Obrázek 1.2.** Načtení údajů po úspěšném přihlášení bez obnovení stránky, pokud je povolen JavaScript.

Ale co když je JavaScript vypnutý? Museli byste zajistit, aby kód jazyka HTML navedl uživatele na samostatnou stránku s přihlašovacím formulářem. Odesláním formuláře pošle uživatel data na server, dojde k obnovení stránky a server rozhodne, jakou stránku zobrazí, a to podle výsledku přihlášení – ať už úspěšném, nebo neúspěšném. V obou případech je funkce stejná, ale liší se svým průběhem.

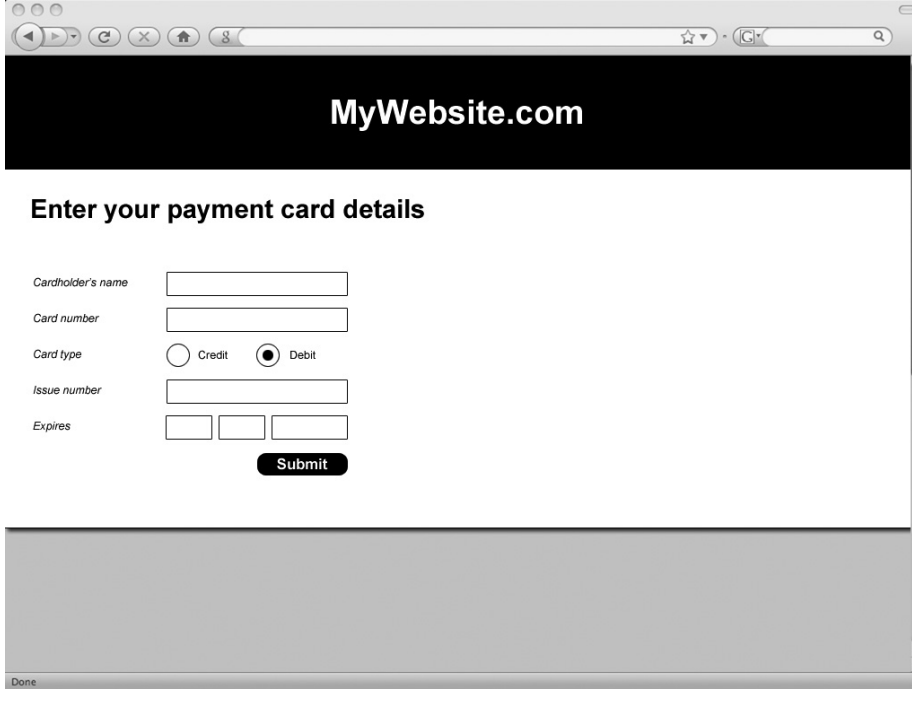
Další příklad – předpokládejme, že máte stránku obsahující formulář, který sbírá informace o platbě pro online rezervační systém. V tomto formuláři by byla pole, která by se zobrazila na základě vybraného typu platební karty. Když by uživatel vybral způsob platby kreditní kartou (jak ukazuje obrázek 1.3), zobrazila by se jiná pole než u platby debetní kartou (obrázek 1.4). Například pole pro číslo vydání budete potřebovat jen u některých debetních karet a naopak datum vydání karty je vhodné jen pro některé kreditní karty. Rovněž budete pravděpodobně chtít zabránit uživateli v tom, aby odeslal nesprávné datum, například 30. února.

K těmto účelům můžete jako weboví vývojáři použít jazyk JavaScript. Jazyk JavaScript spouští události, když uživatele provádí určité akce v prohlížeči a v reakci na tyto události se může provést nějaký zdrojový kód. Dokonce můžete událost zrušit, například když uživatel odešle formulář a vy zjistíte, že data neprošla kontrolou platnosti, můžete odeslání formuláře zrušit.

Pomocí jazyka JavaScript můžete hlídat změnu pole s typem platební karty. Když uživatel vybere nějaký přepínač, spustí se určitá události. V reakci na tyto události můžete provést nějaký kus zdrojového kódu a podle vybraného typu platební karty buď zobrazit, nebo skrýt požadovaná pole.




**Obrázek 1.3.** Formulář platební karty zobrazující pole kreditní karty.




**Obrázek 1.4.** Formulář platební karty zobrazující pole debetní karty.

Rovněž můžete naslouchat události odeslání formulářem, a pokud ji uživatel vyvolá, spustíte malý kousek kódu jazyka JavaScript, který zkontroluje hodnoty polí formuláře. Odeslání formuláře můžete zrušit, pokud se rozhodnete, že data nevyhovují.

Co se stane, když vaše stránky navštíví uživatel s prohlížečem, který nepodporuje jazyk JavaScript? Vybere typ platební karty, ale nic se nezmění. Ve skutečnosti, aby se projevily nějaké změny ve vzhledu stránky, musel by uživatel odeslat data na server a server by provedl podobné zpracování, jaké jste udělali pomocí jazyka JavaScript.

Z pohledu použitelnosti budete pravděpodobně považovat za nevhodné nutit uživatele, aby odeslal formulář, když zvolí typ platební karty. V tomto případě je ideální řešení umístit všechna pole do jediné stránky a dovolit uživateli vyplnit jen ta, která odpovídají jeho typu platební karty. Když nakonec uživatel odešle formulář, zpracuje se na straně serveru a server zkontroluje informace o platební kartě, správnost zadaného data a pokud dojde k chybě, zobrazí stránku s chybovou zprávou.

#### Pojmenovávejte a seskupujte soubory a složky konzistentně

Zaváděním pravidel a konvencí, týkajících se pojmenování složek, souborů a jejich obsahu, usnadňujete sobě a dalším vývojářům prohledávání souborů a zdrojového kódu. Udržovat a měnit zdrojový kód je mnohem jednodušší, pokud existuje konvence pojmenování, která zajišťuje, že programátor vždy ví, jak pojmenovat své prostředky. Dále v této kapitole, v části „Uspořádání složek, souborů a prostředků," najdete několik příkladů uspořádání složek, které byste mohli použít.

#### Udržování čistého jádra zdrojového kódu

Měli byste zajistit, že k projektu patří jen ty soubory, které potřebujete, aby vaše webové stránky fungovaly tak, jak mají – ani málo, ani moc. Časem mohou být některé soubory nahrazeny jinými, stejně jako některá pravidla jazyka CSS a soubory s kódem jazyka JavaScript.

Doporučuji pravidelně mazat všechny nadbytečné sobory, složky a zdrojový kód z jádra zdrojového kódu. Tím snížíte velikost projektu, pomůžete jiným vývojářům snáze proniknout do zdrojového kódu a uživatelé nebudou stahovat soubory, které nikdy nevyužijí.

Abyste předešli problémům, které nastávají, když omylem smažete soubory nebo potřebujete soubory, které jste dříve vymazali, doporučuji používat nějaký systém pro správu zdrojového kódu. Takový systém bude vytvářet zálohy souborů projektu a umožní vám kdykoliv se vrátit ke starší verzi nějakého souboru nebo složky, a to i tehdy, když je z projektu odstraníte. Více informací najdete v pozdější části této kapitoly s názvem „Ukládání souborů: systémy pro správu verzí."

#### Navrhujte efektivní zdrojový kód

Návštěvníci vašich webových stránek očekávají interaktivní uživatelské rozhraní. Pokud uživatelé stisknou tlačítko, očekávají nějakou reakci, aby věděli, že se něco děje.

Zdrojový kód jazyků HTML, CSS a JavaScript je interpretován webovým prohlížečem a plně závisí na schopnostech počítače nebo jiného zařízení, které používá koncový uživatel. Váš zdrojový kód musí být stručný a efektivní, aby se dal rychle stáhnout, správně zobrazit a reagoval svižně. Druhá část této knihy se zaměřuje na výkon a vysvětluje, jak vytvářet lehčí, hubenější a rychlejší zdrojový kód, který koncoví uživatelé ocení.

#### Nepoužívejte technologie bezdůvodně

V široké komunitě webových vývojářů často uslyšíte chválu na nové technologie, které mají nějakým způsobem vylepšit vaše webové stránky. Nedávno byla takto vychvalována technologie asynchronního JavaScriptu a XML (Ajax), která umožňuje jazyku JavaScript komunikovat s webovým serverem, takže není nutné obnovovat stránku tak často. Tuto techniku si weboví vývojáři rychle oblíbili a začali ji používat u každého nového projektu.

Problémem je, že začaly vznikat stránky, které se výhradně spoléhají na Ajax, a proto závisí na jazyku JavaScript. Uživatelé bez podpory jazyka JavaScript ve svých webových prohlížečích, tj. uživatelé některých mobilních webových prohlížečů, uživatelé s omezeným přístupem v pracovním prostředí, postižení uživatelé se speciálními požadavky na webový prohlížeč a externí roboti (například vyhledávací roboti), neměli přístup k informacím, které se běžně zobrazovaly na stránkách HTML. Naopak uživatelé s webovými prohlížeči s podporou jazyka JavaScript často zjišťovali, že pokud zůstanou nějakou dobu na určitých webových stránkách, které se velmi spoléhají na technologii Ajax, jejich webový prohlížeč se zpomalí nebo přestane reagovat. Někteří weboví vývojáři, kteří propadli novému šílenství, zapomněli programovat tak, aby nedocházelo ke kupení nevrácené paměti.

Budujte své webové stránky na dobrých základech a pevných principech, otestujte nové technologie a jejich omezení, než se rozhodnete, že jsou vhodné pro váš projekt. O technologii Ajax se více dozvíte v kapitole 2 a jak se vyhnout nevrácené paměti ve webových prohlížečích v kapitole 4.

# Osvědčené postupy pro značkování: sémantické HTML

Jazyk HTML nebo XHTML tvoří základ každé webové stránky. V podstatě jde o jediné webové standardy, které musí podporovat všechny dostupné webové prohlížeče. Termín *sémantické* zde říká, že elementy by měly označovat význam svého obsahu.

Určitě máte výhodu, pokud znáte co největší množství elementů a atributů jazyka HTML/XHTML. Označujte obsah správným elementem: tabulková data elementy tabulek, nadpisy elementy nadpisů, atd. Čím více významu dáte vašemu obsahu, tím více webových prohlížečů, vyhledávacích robotů a jiných nástrojů správně interpretuje váš obsah.

Doporučuji zahrnout veškerý význam do elementů jazyka HTML, dokonce i tehdy, když pravidly jazyka CSS některé elementy ve webovém prohlížeči skryjete. Dobrou radou je označovat obsah tak, jak byste chtěli, aby zněl, kdyby byl čten nahlas. Představte si, že název elementu je přečten nahlas a následuje obsah tohoto elementu. Tímto způsobem ve skutečnosti funguje většina čteček obrazovky, které zvukově zprostředkovávají obsah lidem se zrakovým postižením.

Představte si, že jste vytvořili webovou stránku pro recenze filmů a chcete zobrazit obrázek, který sděluje, že nějaký film dostal čtyři hvězdičky z pěti. Nyní přemýšlejte, jak byste tuto informaci řekli nahlas – nějak takto: „hodnocení filmu je čtyři z pěti možných hvězdiček." Řekněme, že tento text vložíte do kódu jazyka HTML, kde se k němu dostane každý. Vy ale nechcete, aby se tento text zobrazil na stránce, chcete zobrazit pouze obrázek se čtyřmi hvězdičkami. V tuto chvíli přichází na scénu jazyk CSS. Můžete do elementu obklopujícího tento text přidat atribut `class` a pomocí jazyka CSS definovat, že tato třída má skrytý text a má zobrazit obrázek podle zadaného hodnocení. Pravidla pro skrytí částí textu, která fungují ve všech prohlížečích, včetně čteček obrazovky, najdete v pozděj-

ší části této kapitoly s názvem „Praxí osvědčené postupy pro formátování: CSS.“ Kód jazyka HTML pro danou část stránky by mohl vypadat takto:

```
<div class="rated-four-out-five">
  Hodnocení filmu je čtyři z pěti možných hvězdiček
</div>
```

## Naučte se elementy jazyka HTML

Pokud jste zkušený webový vývojář, který již pracoval na několika webech a označkoval jejich obsah sémanticky, znáte již celou řadu elementů: například `h1`, `h2`, `p`, `img`, `ul` a `li`. Vývojáři si však zřídka pamatují několik méně známým elementů. Bez některých z těchto elementů riskujete, že označkujete vaše dokumenty špatně a přicházíte o příležitost sdělit význam obsahu některým uživatelům, vyhledávacím robotům, atd.

Následuje několik elementů, které vkládají důležitý význam pro prohlížeč nebo koncového uživatele, ale běžně na ně programátoři zapomínají:

- `abbr`: zkratka, používá se pro označení řádkového textu jako zkratky. Spousta prohlížečů po najetí kurzorem myši nad takto označeným textem zobrazí nezkrácenou verzi.

  ```
  <abbr title="et cetera">etc.</abbr>
  ```

- `acronym`: akronym, slouží k označení řádkového textu jako akronymu. Spousta prohlížečů po najetí kurzorem myši nad takto označený text zobrazí delší verzi.

  ```
  <acronym title="World Wide Web">WWW</acronym>
  ```

- `address`: kontaktní informace pro webovou stránku. Na první pohled by se mohlo zdát, že tento element by měl označovat poštovní adresu zobrazenou na stránce. To je však nesprávné použití tohoto elementu. Tento element by měl označovat kontaktní údaje autora stránky. (Poštovní adresa může být samozřejmě součástí těchto údajů.)

  ```
  <address>
    Author: Den Odell<br />
    <a href="mailto:me@denodell.com">Napište autorovi</a>
  </address>
  ```

- `blockquote`: dlouhý citát. Důležitou vlastností blokových citátů, na kterou se často zapomíná, je, že mohou obsahovat pouze blokové elementy. Proto musí být samotný citát nejprve umístěn do odstavce nebo jiného blokového elementu.

  ```
  <blockquote>
    <p>If music be the food of love, play on,<br />
    Give me excess of it, that surfeiting,<br />
    The appetite may sicken, and so die.</p>
  </blockquote>
  ```

- `ins` a `del`: Vložený a smazaný text. Element `del` označuje kus obsahu, který byl vymazán. Element `ins` ukazuje, že kus obsahu byl vložen do stránky. Tyto elementy lze například použít u příspěvků blogu, když se autor po zveřejnění článku rozhodne změnit nějakou větu. Těmito elementy lze sémanticky označit tuto akci. Často bývá text v elementu `del` prohlížeči zobrazován přeškrtnutý.

  ```
  Česká republika má <del>9</del> <ins>10</ins> miliónů obyvatel.
  ```

Nezapomeňte na předchozí elementy, když budete vytvářet své webové stránky. Nepřehlížejte příležitosti, kdy můžete tyto elementy zanést do zdrojového kódu, aby správně označily váš obsah.

**Tip**

Při programování mějte po ruce referenční seznam elementů a atributů a občas se na něj mrkněte. Dobrý zdroj informací o elementech a atributech jazyka XHTML najdete na internetové adrese *http://www.w3schools.com/tags/*.

## Začněte definicí typu dokumentu

Každá stránka HTML nebo XHTML by měla začínat definicí typu dokumentu (DTD). Tato definice by měla předcházet jakémukoliv jinému elementu webové stránky. Definice DTD označuje standard jazyka HTML, který byl použit pro obsah stránky, což je důležitá informace pro jakýkoliv program, který má analyzovat obsah stránky. Pokud například prohlížeč ví, že dokument obsahuje elementy jazyka XHTML, může předpokládat, že každý element je uzavřený, že velikost písmen v názvech elementů a atributů je konzistentní a že elementy jsou správně vnořené, protože tato pravidla platí u dokumentů jazyka XHTML.

Definice DTD není běžný element, a proto nemusí být uzavřený, jak vidíte na tomto příkladu:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Předchozí definice DTD říká, že obsah stránky splňuje specifikaci XHTML 1.0 Transitional a na jaké adrese lze tuto specifikaci najít.

Pokud vynecháte definici DTD, riskujete, že webový prohlížeč zkusí určit použitý standard sám, což může vést k chybnému zobrazení stránky.

#### Přepínání pomocí DOCTYPE

Ke starším verzím prohlížeče IE od firmy Microsoft (do verze 5.5, včetně) měla řada programátorů stížnosti, protože prohlížeč nezobrazoval některé styly v souladu s doporučením konsorcia W3C. Konkrétně – blokový model nesouhlasil s doporučením W3C, ani s implementacemi ostatních prohlížečů. Tento model totiž určuje, jak aplikovat vlastnosti jazyka CSS `width` a `height` na blokové elementy, které mají rovněž vlastnost `padding`.

Programátoři společnosti Microsoft se u vývoje prohlížeče IE 6 dostali do prekérní situace – buď mohli přijmout správnou implementaci a zapříčinit tak, že všechny existující stránky, navržené pro starší verze prohlížeče, se budou zobrazovat špatně, nebo to mohli ponechat ve stejném stavu a donutit programátory používat jiné stylové předpisy pro IE než pro ostatní prohlížeče. Samozřejmě ani jediná varianta nebyla přijatelná. Vymysleli takové řešení, že do nového prohlížeče zabudovali obě metody a vytvořili způsob, jak mezi nimi přepínat pomocí definice DTD.

Napsáním definice DTD bez části s adresou URL uvede vývojář prohlížeč do *režimu quirks*, což je původní, nesprávný způsob zobrazení blokového modelu v IE. Napsáním definice DTD s kompletní částí s adresou URL přepne vývojář prohlížeč do *režimu standards*, který odpovídá standardům konsorcia W3C. Volba zobrazovací metody byla tedy ponechána na vývojáři.

#### Výběr definice DTD

Jako vývojáři si jistě přejete, aby webové stránky přijaly všechny standardy, které se týkají zdrojového kódu i softwaru, který jej interpretuje. Musíte si však uvědomit, že začít používat nejnovější doporučení konsorcia W3C nemusí být vždy nejlepší tah. Musíte vzít v úvahu, kolik současných oblíbených webových prohlížečů toto doporučení podporuje.

V době, kdy vyšla tato kniha, měly následující definice DTD plnou podporu v prohlížečích a jsou doporučované:

- HTML 4.01 Strict,
- HTML 4.01 Transitional,
- HTML 4.01 Frameset,
- XHTML 1.0 Strict,
- XHTML 1.0 Transitional,
- XHTML 1.0 Frameset.

Jazyk HTML 4.01 je ustupující verze jazyka HTML 4, která upřednostňuje strukturu nad prezentací. HTML 4.01 Strict by měla být výchozí definicí DTD, pokud tedy neexistuje dobrý důvod, proč použít jinou verzi. Tato definice DTD zobrazuje dokument tím nejstriktnějším způsobem a zaručuje největší kompatibilitu.

HTML 4.01 Transitional obsahuje všechny elementy a atributy verze HTML 4.01 Strict, navíc přidává prezentační atributy, zastaralé elementy a cíle odkazů. HTML 4.01 Transitional byste měli použít, pokud potřebujete ke stránkám přidat starší zdrojový kód.

HTML 4.01 Frameset obsahuje všechny elementy a atributy verze HTML 4.01 Transitional a přidává podporu rámců. Tuto verzi byste neměli používat, dokud není použití rámců nutné.

Jak již bylo řečeno, jazyk XHTML je v podstatě jazyk HTML ve formátu jazyka XML. Analyzátor jazyka XML tedy umí dokument XHTML přečíst a převést ho pomocí jazyka XSLT (Extensible Stylesheet Language Transformations) do libovolné jiné textové podoby.

#### Validace podle definice DTD

To, že vytvoříte definici DTD, ještě neznamená, že váš zdrojový kód splňuje specifikaci určenou touto definicí. Chybám se můžete vyhnout tak, že zdrojový kód zkontrolujete validátorem HTML, který ověří, zda vaše stránka splňuje uvedenou definici DTD.

Jedním z nejlepších validátorů je online nástroj konsorcia W3C na internetové adrese *http://validator.w3.org/*. Validaci můžete provést zadáním veřejné URL adresy, nahráním dokumentu HTML na server nebo přímo vložením zdrojového kódu do textového pole. Klepnutím na tlačítko *Check* zahájíte validaci. Případné chyby budou zobrazeny v pořadí, v jakém se vyskytují ve zdrojovém kódu.

## Jak přidat X před HTML?

Jazyk XHTML je v podstatě jazyk HTML splňující navíc pravidla jazyka XML:

- Všechny elementy musí být dobře formované.

- Všechny názvy elementů a atributů by měly být zapsány buď malými, nebo velkými písmeny, protože v jazyku XML záleží na velikosti písmen. Mnoha lidem se zdá, že malá písmena jsou přehlednější.
- Atributy musí mít uvedené hodnoty, a tyto hodnoty musí být zapsány do uvozovek. Dokonce i jednoduché číselné hodnoty musí být zapsány do uvozovek. V případě, kdy atribut představuje pravdivostní hodnotu (třeba atributy `checked` a `disabled` v elementu `input`), byste měli jako hodnotu zapsat název tohoto atributu, například takto:

```
<input checked="checked" type="checkbox" name="item1" id="item1"
       value="1" />
```

Na rozdíl od jazyka HTML je jazyk XHTML dobře formovaný – každý element, který otevřete, musíte uzavřít. Díky tomu prohlížeč jednodušeji analyzuje kód jazyka XHTML, než kód jazyka HTML, a to je vhodné i pro mobilní aplikace, kde jsou procesory pomalé a paměti není mnoho. Pro mobilní zařízení existuje také varianta XHTML Mobile Profile (v současnosti známá pod označením XTHML Basic), což je podmnožina jazyka XHTML.

Ve skutečnosti je lepší čitelnost XHTML výhodou i pro další webové služby a programy. Díky tomu je jazyk XHTML všestranný a doporučovaný. Všechny příklady ve zbytku této knihy budou upřednostňovat jazyk XHTML před jazykem HTML.

#### Dobře formované dokumenty

Dokument XHTML je dobře formovaný, když všechny jeho elementy mají uzavírací elementy nebo jsou samouzavírací a všechny elementy jsou správně vnořené vzhledem k ostatním. To znamená, že každý element musí být uzavřen, než je uzavřen jeho rodičovský element.

Zde je příklad správně vnořených elementů:

```
<p>
  Toto je odstavec se <em>zdůrazněným</em> textem.
</p>
```

A zde je příklad nesprávného vnoření:

```
<p>
  Toto je odstavec se <em>zdůrazněným</p> textem.
</em>
```

V dokumentech XHML musí být všechny elementy bez vnitřního obsahu samouzavírací, jsou jimi například `<br />` nebo `<img />`. Před poslední znak názvu elementu a lomítko na jeho konci byste měli vložit mezeru. Bez této mezery měly některé starší prohlížeče, včetně prohlížeče Netscape 4, potíže se zobrazením dokumentu.

#### Omezení elementů

Jazyk XHTML klade omezení na vnořování elementů. Například není možné umístit řádkové elementy přímo do elementu `body`, ale musí být vnořené do blokových elementů. Blokové elementy nelze vkládat do řádkových elementů. Některé elementy mohou být řádkové i blokové, v závislosti na jejich obsahu, jsou to třeba elementy `ins` a `del`. Měli byste si tyto elementy vyhledat ve vaší referenční příručce jazyka HTML. Dávejte pozor na elementy, které nemohou obsahovat další elementy, jako například elementy v tabulce 1.1.

**Tabulka 1.1.** Několik elementů s omezením na obsah.

| Element | Omezení |
|---|---|
| `a` | Nemůže obsahovat další elementy `a`. |
| `pre` | Nemůže obsahovat elementy `img`, `object`, `big`, `small`, `sub` a `sup`. |
| `button` | Nemůže obsahovat elementy `input`, `select`, `textarea`, `label`, `button`, `form`, `fieldset`, `iframe` a `isindex`. |
| `label` | Nemůže obsahovat další elementy `label`. |
| `form` | Nemůže obsahovat další elementy `form`. |

## Uvedení praxí osvědčených postupů do praxe

Představte si, že sedíte před vaším počítačem a chystáte se psát dokument HTML podle praxí osvědčených postupů. Jak to uděláte?

Zapamatujte si, že vaším cílem je napsat konzistentní zdrojový kód, který je snadno čitelný pro ostatní vývojáře, a jehož výsledkem bude správně označkovaná webová stránka, která je přístupná koncovým uživatelům. Projděme si jednotlivá pravidla, která by vám měla pomoct s realizací.

### Pište uspořádaný a čistý zdrojový kód

Odsazení zdrojového kódu zlepšuje jeho čitelnost. Díky tomu, že části kódu odsadíte podle příslušné skupiny nebo stupně zanoření, bude zdrojový kód čitelnější a pro ostatní vývojáře snáze pochopitelný. Nezapomeňte, že jedním z cílů je udržovatelnost kódu ostatními programátory. Stejně tak jako uklízíte pokoj, když čekáte návštěvu, měli byste uklidit svůj zdrojový kód, když pravděpodobně budete mít návštěvníky.

Místo mezery byste měli pro odsazení používat tabulátor. To usnadňuje údržbu, zlepšuje čitelnost a snižuje celkovou velikost webové stránky. Ve svém vývojovém prostředí si můžete nastavit, že tabulátor bude odpovídat určitému počtu mezer. Pro čitelnost jsou obvykle dostatečné dvě, čtyři nebo osm mezer.

Bloky zdrojového kódu, které jsou zanořené uvnitř jiných elementů, by měly být odsazené. U každé úrovně zanoření by měl přibýt jeden tabulátor. Elementy, které obsahují jen text, nebo řádkové elementy, nemusí mít svůj obsah odsazený vzhledem k rodičovskému elementu. Prohlédněte si následující příklad, který ukazuje typické odsazení zdrojového kódu.

```
<div id="container">
  <p>Odstavec textu.</p>
  <table>
    <tr>
      <th>Název</th>
      <th>Hodnota</th>
    </tr>
    <tr>
      <td>Červená</td>
      <td>foo</td>
    </tr>
  </table>
</div>
```

#### Omezte počet komentářů

Jsem si jistý, že jste viděli spoustu příkladů dokumentů HTML s komentáři. Komentáře se zapisují takto:

```
<!-- zde je komentář -->
```

Komentáře občas popisují začátek a konec nějakých elementů nebo částí stránky. Přestože to může být užitečné k vyznačení toho, co která část serverem vygenerovaného kódu dělá, většina vývojových prostředí obsahuje nástroje, které umožňují snadno vyhledat začátek a konec částí obsahu, a proto jsou tyto komentáře nadbytečné.

Spousta komentářů v dokumentu HTML znamená, že koncoví uživatelé musí stahovat přes síť více dat, než se jim dokument zobrazí v prohlížeči. Doporučuji vynechat tyto komentáře, ale s následujícími výjimkami:

- Když se může jinému vývojáři zdát, že jste určitý element použili zbytečně. Komentář může podat vysvětlení a předejít nedorozumění.
- Když tyto komentáře mají v určitém prohlížeči nějakou funkci. To je případ podmínkových komentářů v prohlížeči IE, se kterými se seznámíte v následující části kapitoly.

#### V IE používejte podmínkové komentáře

Do prohlížeče IE 5 a novějších (jen pro operační systém Windows) přidala společnost Microsoft velmi užitečnou funkci s názvem *podmínkové komentáře*. Jejich význam spočívá v tom, že když vývojář potřebuje napsat zdrojový kód jen pro určitou nebo kteroukoli verzi prohlížeče IE, použije k tomu speciálně zapsaný komentář a nemusí jazykem JavaScript nebo na serveru detekovat typ prohlížeče. Všem ostatním prohlížečům se obsah tohoto elementu jeví jako standardní komentář, a proto jej ignorují.

Zde je příklad podmínkového komentáře určeného prohlížeči IE 6 a novějším:

```
<!--[if gte IE 6]>
  <p>Serfujete s prohlížečem Internet Explorer 6 nebo novějším.</p>
<![endif]-->
```

Následuje podmínkový komentář určený jen uživatelům prohlížeče IE:

```
<!--[if IE]>
  <p>Používáte prohlížeč Internet Explorer.</p>
<![endif]-->
```

Poslední příklad ukazuje podmínkový komentář pro starší verze IE než verze 7:

```
<!--[if lt IE 7]>
  <p>Používáte prohlížeč Internet Explorer ve verzi starší než verze 7.</p>
<![endif]-->
```

V podmínkových komentářích výraz `lt` (less than) označuje méně než, `gt` (greater than) znamená více než, `lte` (less than or equal to) znamená méně než nebo stejně a `gte` (greater than or equal to) znamená více než nebo stejně.

Tato technika je užitečná zejména tehdy, když importujete externí šablony stylů. Prohlédněte si následující zdrojový kód, který by měl být součástí bloku `<head>` dokumentu XHTML. Tento kód nahraje externí šablonu stylů jen pro uživatele prohlížeče IE 6.

```
<link rel="stylesheet" href="master.css" type="text/css" />
<!--[if IE 6]>
  <link rel="stylesheet" href="master.ie6.css" type="text/css" />
<![endif]-->
```

Většina webových prohlížečů uvidí jen jednu šablonu stylů, `master.css`, následovanou komentářem (obsah obklopený značkami `<!--` a `-->`), který ignorují, jedinou výjimku bude prohlížeč IE 6. Protože IE 6 pozná podmínkový komentář, který označuje, že jeho obsah by měla přečíst jen určitá verze IE, analyzuje zdrojový kód uvnitř tohoto komentáře.

To umožňuje vývojářům udržovat hlavní šablonu stylů pro všechny webové prohlížeče, které se řídí standardy. Ale když něco nefunguje pro určité verze prohlížeče IE, mohou vývojáři vložit menší šablonu stylů, která opravuje špatně zobrazené elementy. V budoucnosti, až bude IE 6 jen pouhou vzpomínkou, mohou vývojáři podmínkové komentáře z kódu odstranit.

#### Nastavte element html správně

Ve všech dokumentech HTML následuje za definicí DTD element `html`, který obaluje zbylý obsah dokumentu.

U dokumentů HTML můžete tento element ponechat bez atributů, ale v dokumentech XHTML musí obsahovat několik atributů. Prohlédněte si tento příklad z dokumentu XHTML:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="cs" lang="cs"
      dir="ltr">
```

Tento příklad se skládá z následujících součástí:

- `xmlns`: Tento atribut definuje obor názvů pro vlastní elementy dokumentu (je povinný, ačkoliv se s ním ve skutečných aplikacích setkáte v současnosti zřídka).
- `xml:lang` a `lang`: Tyto atributy specifikují jazyk, například `cs` nastavuje češtinu pro celý dokument. Tam, kde se mění jazyk obsahu v dokumentu, například je použit francouzský výraz *c'est la vie*, musíte text označit elementem. Ve většině případů byste k elementu `span` přidali atributy `xml:lang` a `lang`. U předchozího příkladu byste jako hodnotu atributu dali `fr-FR`.
- `dir`: Tento atribut definuje směr čtení obsahu dokumentu. Pro většinu západních jazyků by hodnotou bylo `ltr` (left to right) označující směr zleva doprava. Pokud se však v obsahu mění směr, musíte vložit atribut `dir` k elementu, který obklopuje takový obsah. Jedinou možnou alternativou je hodnota `rtl` označující směr zprava doleva.

#### Uveďte typ obsahu

Je vhodné říct prohlížeči nebo specializovanému programu, jaký je typ obsahu dokumentu, pokud to není možné zjistit ze samotného souboru. Snižuje to pravděpodobnost omylu a zaručuje, že obsah bude analyzován tak, jak bylo myšleno. Za tímto účelem vložte element `meta` do části `<head>`, jak vidíte v tomto příkladu:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

Tímto řádkem kódu sdělujete prohlížeči, že obsah používá znakovou sadu UTF-8. Tato znaková sada je velmi flexibilní, vystačí pro většinu světových jazyků, aniž byste museli převádět kódování znaků v dokumentu.

#### Nastavte titulek stránky

Element `title` je základní součást každého dokumentu. Měl by obsahovat jednoduchý nadpis, který popisuje daný dokument co nejmenším počtem slov. Vyhledávače jej používají jako text odkazu na vaše webové stránky ve výsledcích vyhledávání, dále se zobrazuje v záhlaví okna webového prohlížeče.

Když to víte, pravděpodobně bude chtít do titulku přidat informaci o tom, kde se daný dokument nachází ve struktuře vašich webových stránek, abyste usnadnili orientaci uživatelům, kteří na daný dokument přejdou z výsledků nějakého vyhledávače. Následující formát titulku se zdá vhodný – poskytuje tuto informaci a zároveň je čitelný:

```
<title>Titulek stránky - Název části - Název webu</title>
```

*Titulek stránky* z tohoto příkladu se bude téměř vždy shodovat s hlavním nadpisem stránky, který je uveden uvnitř elementu `h1`. Samozřejmě, že mezi hodnotami lze použít libovolný oddělovač, důležité je pořadí.

Je logické, že každá stránka vašich webových stránek by měla mít jedinečný titulek stránky v rámci dané části, aby se ve výsledcích vyhledávačů nebo v mapě webu nezobrazovaly duplicitní názvy.

#### Oddělujte prezentaci, obsah a chování

Je důležité oddělovat obsah dokumentu od kódu, který má na starost návrh a rozvržení. Díky tomuto oddělní můžete snadno vyměňovat styly, aniž byste museli měnit kód jazyka HTML. Dokonce můžete snadno vyměnit rozvržení webových stránek a nemusíte měnit jejich obsah.

Stylům zapsaným přímo do kódu jazyka HTML (do atributu `style`) byste se měli vyhnout, protože kvůli nim si neuvěřitelně komplikujete údržbu. Vývojáři by měli do souborů s šablonami stylů vkládat vše, co se týká vzhledu a rozvržení a do dokumentů HTML vše, co se týká obsahu. Tyto dvě věci by se neměly nikdy míchat. Rovněž byste neměli do dokumentů HTML vkládat žádný kód jazyka JavaScript.

Místo toho se v dokumentu HTML odkažte na šablonu stylů pomocí elementu `link` a na zdrojový kód jazyka JavaScript pomocí elementu `script`. Atributy `class` a `id` elementů HTML dokumentu by měly být jediným pojítkem s těmito externími soubory, do dokumentů byste neměli vkládat atribut `style` a atributy pro obsluhu událostí.

Je možné se na soubory s šablonami stylů odkazovat podle zařízení, na kterém bude dokument zobrazen. Například budete pravděpodobně chtít použít samostatné stylové předpisy pro tiskárnu a obrazovku, jak vidíte zde:

```
<link rel="stylesheet" href="master.css" media="screen" />
<link rel="stylesheet" href="master-print.css" media="print" />
```

Když element `link` obsahuje atribut `media`, tiskárna načte tuto šablonu stylů jen v případě, že daný atribut obsahuje hodnotu `print` a obrazovka načte tuto šablonu stylů, jen když má daný atribut hodnotu `screen`. Díky tomu můžete použít jiný vzhled pro jiné prezentační zařízení. Například na tiskárně budete pravděpodobně potřebovat jen samostatný obsah. Můžete vytvořit styly, které skryjí navigaci, záhlaví a zápatí stránky a ponechají jen hlavní obsah, který obvykle lidé chtějí vytisknout a přečíst, pokud právě nemají po ruce webový prohlížeč.

#### Přidejte element, který obalí obsah celého dokumentu

Uvnitř elementu `body` budete pravděpodobně chtít přidat styly s rozvržením celé stránky. Brzo zjistíte, že aplikováním těchto stylů na element `body` nedosáhnete všech možných pozic a rozvržení, jaké byste chtěli. Doporučuji používat element, nejčastěji `div`, který obalí veškerý obsah stránky a aplikovat styly rozvržení na tento element, a ne na samotný element `body`.

Element `div` definuje pouze blok obsahu a nepřidává žádný význam tomuto obsahu. Pomocí atributu `id` si tento element pojmenujte, abyste se na něj mohli odkázat ve stylech jazyka CSS:

```
<body>
  <div id="page">
  ...
  </div>
</body>
```

Pokud máte jednoduchý design stránek, možná si vystačíte jen s elementem `body`. Avšak i v tomto případě zvažte, zda nepoužijete obalový element, protože nikdy nevíte, jak se může design změnit v budoucnu.

#### Pomozte jazykům CSS a JavaScript identifikovat jednotlivé stránky

Zastávám názor, že element `body` každé stránky by měl mít unikátní atribut `id`. Představte si, že budete mít obecné styly pro webové stránky s velkým počtem stránek a jednu určitou stránku budete chtít zobrazit odlišně. V tomto případě budete muset vytvořit nějakou výjimku zobrazení. Pokud má každá stránka jedinečný atribut `id`, můžete vytvořit styly, které se aplikují pouze na jednu vybranou stránku.

Představte si, že budete chtít, aby odstavec obsahoval černý text na bílém pozadí, ale na domovské stránce to budete chtít obráceně – bílý text na černém pozadí. Domovská stránka bude mít jedinečný atribut `id`:

```
<body id="homepage">
```

a styly definující text odstavce by mohly vypadat takto:

```
p {
  background: white;
  color: black;
}

#homepage p {
  background: black;
  color: white;
}
```

Kromě toho můžete díky jedinečnému atributu `id` vyhledat určitou stránku pomocí jazyka JavaScript, a to metodou `document.getElementById`:

```
var homepage = document.getElementById("homepage");
```

#### Pojmenovávejte identifikátory a třídy konzistentně

Každý logický blok byste měli umístit do vlastního elementu `div` se smysluplnou hodnotou atributu `id`. Některé komponenty jsou společné pro většinu stránek – například navigace, záhlaví a zápatí. Tyto komponenty by měly být konzistentní mezi stránkami a projekty, protože můžou pomoct

s budoucí údržbou. Zvažte, zda byste neměli použít následující hodnoty atributu `id` pro příslušné komponenty:

```
<div id="header">
<div id="content">
<div id="aside">
<div id="footer">
<div id="navigation">
```

Hodnoty atributů `id` a `class` byste měli volit podle toho, co dané elementy obsahují, a ne podle toho, jak vypadají. Například tento zápis:

```
<p class="error">
```

je lepší než tento:

```
<p class="red-text">
```

**Tip**

Pokud oddělíte slova uvnitř hodnot atributů `id` a `class` pomlčkou (`-`), budou čitelnější.

Pokud označíte atributy `class` informací o obsahu, nebudete muset měnit kód HTML, pokud návrháři změní některé části stránky, změníte jen šablonu stylů. To znamená, že dokument HTML popisuje pouze svůj obsah.

#### Uspořádejte správně obsah

Uspořádat správně obsah dokumentu je stejně tak důležité jako označit jej správnými elementy. Nezapomeňte, že když bude stránka čtena nahlas, nejdůležitější obsah musí být přečten první a nejméně důležitý obsah, například navigace nebo copyright, jako poslední. Čtečky obrazovky předčítají obsah dokumentu přesně tak, jak je zapsán ve zdrojovém kódu, to znamená, že to, co je napsané první, přečtou jako první.

Aby uživatelé čteček obrazovky získali smysluplné informace, měli byste obsah elementu `body` uspořádat následovně:

- titulek stránky,
- krátký seznam odkazů, které umožní čtenáři rychle přejít na určitou část dané stránky,
- hlavní obsah dokumentu,
- doprovodný obsah (postranní panel, související informace a odkazy),
- hlavní navigace,
- zápatí a copyright.

Pomocí jazyka CSS můžete měnit rozvržení stránky, aby odpovídalo aktuálnímu designu. To bude vysvětleno později v této kapitole, v části „Pravidla přístupnosti pro styly."

#### Oddělte obrázky popředí od pozadí

Je důležité oddělit obrázky, které se vztahují k obsahu, od těch, které jsou součástí designu.

Obrázky vztahující se přímo k obsahu stránky, ilustrace, grafy, obrázky domácích mazlíčků, atd., byste měli označovat elementem `img`. Na všechny ostatní obrázky, včetně log a ikonek společnosti,

byste se měli odkazovat v souborech s šablonou stylů jako na obrázky pozadí. Díky tomu budete mít další možnost, jak rozlišit mezi rozvržením stránky a obsahem.

Položte si otázku: „Když vytisknu obsah stránky, bude tam ten obrázek k něčemu, vztahuje se k obsahu?“ Brzo zjistíte, že většinu obrázků budete umísťovat do šablon stylů, a ne do elementů `img`.

Tam, kde používáte element `img`, nezapomeňte uvést atribut `alt` k popisu obrázku pro uživatele, kteří si jej z nějakého důvodu nemohou prohlédnout. U složitějších obrázků, jako jsou grafy nebo diagramy, bude vhodný detailnější popis. V takovém případě uvažujte, zda nepoužijete atribut `longdesc`, který obsahuje URL stránky s detailním popisem daného obrázku.

#### Používejte tabulky správným způsobem

Ještě před několika lety bylo zcela obvyklé rozmísťovat komponenty stránek pomocí elementů jazyka HTML určených pro tabulky. Element `table` byste měli používat jen k reprezentaci tabulkových dat, a ne pro rozmísťování obsahu stránky, pro tento účel je vhodnější jazyk CSS.

Existuje několik způsobů, jak přidat další sémantickou informaci k tabulkovým datům, samozřejmě vylepšují přístupnost informací v tabulkách, a proto si zaslouží pozornost:

- Bezprostředně za začátek elementu `table` umístěte element `caption`, tím tabulku pojmenujete. Nezapomeňte, že pomocí šablon stylů můžete obsah tohoto elementu kdykoliv schovat, pokud jej nechcete zobrazit, ale je vhodné, abyste do vašich stránek přidali co nejvíce sémantických dat.
- Atributem `summary` elementu `table` poskytněte stručný přehled obsahu tabulky a její účel. Obsah tohoto atributu mohou čtečky obrazovky číst nahlas.
- Seskupujte záhlaví, tělo a zápatí tabulky pomocí elementů `thead`, `tbody` a `tfoot`. Dávejte pozor, protože tyto elementy musí být umístěny v přesném pořadí – nejprve `thead` a `tfoot` a potom `tbody`. Díky tomu může prohlížeč zobrazit řádky záhlaví a zápatí, zatímco se načítá obsah těla tabulky.
- Buňky záhlaví označujte elementy `th` a buňky s daty elementy `td`.
- Každé buňce záhlaví dejte jedinečnou hodnotu atributu `id`. Všem buňkám s daty přiřaďte jako hodnotu atributu `headers` seznam hodnot atributů `id` příslušných buněk záhlaví, které budou oddělené čárkami.

Následuje příklad tabulky se všemi těmito elementy:

```
<table summary="Tabulka ukazující, že průměrný věk studentů tohoto kurzu je 26">
  <caption>Tabulka studentů registrovaných do tohoto kurzu a jejich věk</caption>
  <thead>
    <tr>
      <th id="student-name">Jméno studenta</th>
      <th id="age">Věk</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <th id="average-age">Průměrný věk</th>
      <td headers="age, average-age">26</td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td headers="student-name">Jan Novák</td>
```

```
      <td headers="age">24</td>
    </tr>
    ...
    <tr>
      <td headers="student-name">Petr Kolář</td>
      <td headers="age">28</td>
    </tr>
  </tbody>
</table>
```

#### Vylepšete své formuláře

Logicky související části formuláře byste měli seskupovat elementem `fieldset`, každý takový element obsahuje jeden element `legend` s nadpisem dané skupiny polí formuláře. Například formulář aplikace pro práci s kreditní kartou by mohl mít části pro osobní údaje, historii plateb a bankovní údaje. V tomto případě by formulář měl tři elementy `fieldset` s elementy `legend`, které by obsahovaly texty `Osobní údaje`, `Historie plateb` a `Bankovní údaje`, jak vidíte zde:

```
<form method="get" action="/">
  <fieldset>
    <legend>Osobní údaje</legend>
    ...
  </fieldset>

  <fieldset>
    <legend>historie plateb</legend>
    ...
  </fieldset>

  <fieldset>
    <legend>Bankovní údaje</legend>
    ...
  </fieldset>

  <input type="submit" value="Uložit" />
</form>
```

Ujistěte se, že každé pole formuláře má přiřazený textový popisek pomocí elementu `label`. Tyto elementy mají atribut `for`, který by měl obsahovat stejnou hodnotu jako atribut `id` souvisejícího pole. Webové prohlížeče většinou po klepnutí na obsah elementu `label` umístí kurzor do příslušného pole a umožňují tak uživateli snadněji pracovat s tímto polem. Abyste mohli jednoduše aplikovat styly na pole formulářů a související popisky, obalte pole a jeho popisek elementem `div`. Zde je příklad:

```
<div class="field">
  <label for="first-name">Jméno</label>
  <input type="text" id="first-name" name="first-name"/>
</div>
```

Věnujte pozornost formulaci textů uvnitř elementů `legend` a `label`, které jsou umístěné v elementu `fieldset`. Když tyto popisky čte nahlas čtečka obrazovky, přečte obsah elementu `legend` před obsahem každého elementu `label`. Musíte se ujistit, že toto sdělení bude dávat posluchači smysl.

#### Nepoužívejte rámce

Měli byste se vyhýbat rámcům za každou cenu. Kvůli nim se obsah a rozvržení webových stránek špatně udržuje, nevyhovují zařízením s malými obrazovkami (jako jsou mobilní telefony) a znepříjemňují navigaci v obsahu uživatelům, kteří nepoužívají myš jako primární typ vstupního zařízení.

Řádkové rámce (element `iframe`) byste měli používat s rozvahou, pokud vůbec. Mohou zmást koncové uživatele a schází jim základní přístupná povaha jednostránkového dokumentu HTML.

## Pravidla přístupnosti pro webový obsah

Pokud neznáte doporučení WCAG (Web Content Accessibility Guideline), můžete si tato doporučení konsorcia W3C přečíst na internetové adrese *http://www.w3.org/TR/WCAG20/*. Tato pravidla stanovují tři úrovně přístupnosti webové stránky: A, AA a AAA, kde AAA je nejpřístupnější obsah.

Pravidla přístupnosti nespoléhají jen na technologii, ale také na kreativní design, copywriting a architekturu informací. Kompatibilita AAA je svatým grálem každého vývojáře webových stránek a vždy byste se o ní měli snažit. Kompatibilita AA je však většinou dostačující, pokud je vývojář tlačen časem nebo požadavky na design. Následující pravidla v této kapitole vás navedou na cestu k tvorbě maximálně přístupných stránek, ale musíte si uvědomit, jak jednotliví uživatelé komunikují s webovými stránkami, proto doporučuji přečíst dokument WCAG.

#### Nenechte se zmást přístupovými klávesami

Přístupové klávesy jsou klávesové zkratky, které umožňují uživatelům přeskočit na určitý obsah stránky nebo externích stránek. Dříve byly považovány za skvělý nástroj, ale moderní trendy je zavrhly a vy byste se měli přístupovým klávesám vyhnout.

Klávesové zkratky jsou velmi užitečné pro koncové uživatele, kteří nepoužívají myš. Tito uživatelé mají však nastavené vlastní klávesové zkratky v operačním systému. Bohužel přístupové klávesy často kolidují s uživatelem definovanými klávesovými zkratkami. Uživatelé budou zmatení, když si budou myslet, že používají své klávesové zkratky, ale ve skutečnosti budou používat přístupové klávesy a naopak.

Přístupové klávesy mohou také zmást z toho důvodu, že různé webové stránky používají různé klávesy k vykonání stejné akce. To znamená, že uživatelé si musí pamatovat novou sadu klávesových zkratek pro každou webovou stránku, kterou navštíví. To není zcela obvykle řešení ve světě výpočetní techniky, kde většinou určité klávesové zkratky provádí stejnou akci bez ohledu na to, ve kterém jste programu.

Jedním zlatým pravidlem přístupnosti je snižovat zmatení mezi koncovými uživateli. Proto jsou přístupové zkratky teoreticky skvělé, ale prakticky k ničemu.

#### Nenechte se zmást tabelačními indexy

Dalším takzvaným vylepšením přístupnosti, kterému byste se měli vyhýbat, jsou tabelační indexy. Ty určují pořadí, ve kterém dostávají vstup od uživatele odkazy, pole formulářů, atd., pokud uživatel stiskne tabulátor na klávesnici. Bohužel není možné nastavit tabelační index jednomu poli. Pokud nastavíte tabelační index jednomu elementu, musíte jej nastavit všem, v opačném případě nemáte kontrolu nad tabelačním pořadím. Obtížná údržba tabelačních indexů a jejich mizivý dopad na přístupnost má za následek, že se příliš nepoužívají.

Lepším řešením je dobře strukturovaný dokument, který má důležitý obsah na začátku a odkazy a pole formulářů uspořádané ve zdrojovém kódu tak, jak chcete, aby se zaměřovaly po stisku tabulátoru. Potom můžete pomocí stylů umístit tyto elementy na obrazovce, jak uznáte za vhodné.

#### Nespoléhejte se na zásuvné moduly

Nemůžete předpokládat, že bude zásuvný modul obecně v prohlížeči dostupný. Jelikož není součástí prohlížeče, nelze jeho přítomnost zaručit. Jestliže koncový uživatel tento zásuvný modul nemá, například Adobe's Flash Player, měli byste mu poskytnout alternativní obsah, jak bylo vysvětleno dříve.

Není moudré implementovat celou webovou stránku jako filmový klip Flash, ačkoliv se s tím na webu setkáte poměrně často. Obsah animace zůstává nedostupný uživatelům s určitými prohlížeči nebo bez daného zásuvného modulu a nevidí jej ani většina vyhledávačů. Místo toho uvažujte, zda jen některé komponenty vaši stránky neimplementovat jako animaci – například přehrávání filmové ukázky uvnitř animačního okénka nebo přidání kreativních efektů k navigačnímu panelu. Následně zajistěte, že uživatelé bez tohoto modulu budou moct daný obsah získat jinou cestou. Například nabídnete odkaz na stažení filmové ukázky nebo čistou navigaci bez animací. Tímto způsobem vytvoříte pěkné a chytré webové stránky, které neobětují vzhledu přístupnost.

#### Přidávejte sémantické informace, kam můžete

Objevuje se několik otevřených standardů, které přidávají další význam určitým blokům dokumentu HTML. Jedním takovým standardem jsou *mikroformáty*. Mikroformáty jsou bloky dokumentu HTML, které reprezentují lidi, události, elementy, atd. Jsou čitelné lidmi i stroji a zaznamenají více dat, než když přiřadíte hodnoty určitým atributům.

Například mikroformát hCard slouží k sémantickému zápisu vizitek ve formátu vCard, který se běžně používá pro výměnu vizitek mezi počítačovými programy. Příklad zápisu jména, URL webových stránek a telefonního čísla mobilního telefonu pomocí mikroformátu hCard:

```
<div class="vcard">
  <a class="url fn" href="http://www.denodell.com">
    Den Odell
  </a>
  <span class="tel">
    <span class="type">Mobile</span>
    <span class="value">+441234567890</span>
  </span>
</div>
```

Jaké použijete elementy, je nepodstatné. Důležitější jsou hodnoty atributů, které rozpoznávají různé programy k tomu určené, včetně několika současných webových prohlížečů. Pomocí tabulek stylů lze schovat jakoukoliv část obsahu, kterou nechcete zobrazit na vaší stránce, ale sémantická data zůstanou ve zdrojovém kódu.

**Tip**

Existují další mikroformáty pro označení událostí (hCalendar), recenzí (hReview) a jiné. Stále však vznikají nové mikroformáty, proto sledujte vývoj a nové příklady na internetové adrese *http://microformats.org/*.

# Praxí osvědčené postupy pro formátování: CSS

Jako programátoři jste již určitě někdy otevřeli soubor předpisem šablonou stylů, který vytvořil někdo jiný, a pomysleli jste si: „Jak mám tomu rozumět?“ Jakmile oddělíte všechny styly rozvržení od dokumentu, všimnete si, kolik definic stylů tvoří jedinou stránku.

Protože pro rozvržení stránky je potřeba ohromné množství definic stylů, stane se časem nemožné udržovat rozvržení stránek bez nějakých pravidel na strukturování souborů s šablonami stylů v jazyku CSS.

## Snaha o reprodukci designu s přesností na pixel

Pravděpodobně se budete vždy snažit kopírovat kreativní design tak přesně, jak to jen bude možné. Někdy však tato pixelová přesnost nestojí za to, co touto snahou vizuálně získáte.

Představte si třeba pole formulářů. Měl jsem k dispozici spoustu kreativních designů, které se snažily měnit vzhled šipky v pravé části rozevíracího seznamu (element `select`). Ti z nás, kteří se někdy snažili aplikovat na tento seznam styly, objevili extrémní rozdíly mezi jednotlivými prohlížeči.

Zdá se, že čím modernější webový prohlížeč, tím větší kontrolu nad běžnými ovládacími prvky máte. To samozřejmě většině z vás nepomůže. Spousta z vás se jistě snaží o kompatibilitu s prohlížeči typu IE 6, bez ohledu na to, jak archaicky jejich formulářové prvky vypadají.

Řešením je přístup maximální snahy. U těch prohlížečů, které to podporují, poskytněte formulářové prvky s vlastním vzhledem. U starších prohlížečů, se kterými nedosáhnete požadovaných výsledků, musíte být střízliví.

Všechny hlavní prohlížeče vám umožní změnit barvu pozadí, barvu textu, písmo a výšku řádku (specifikace výšky může mít jiný výsledek v různých prohlížečích, proto je potřeba trochu experimentovat). Ne všechny prohlížeče vám umožní změnit styl a barvu okrajů nebo upravit vzhled šipky u rozevíracího seznamu. Hlavně musíte zajistit, aby si toho byl vědom váš zákazník. Jinak budete muset upravit design tak, aby vypadal stejně ve všech prohlížečích.

Ze zkušenosti a díky zdravému rozumu víme, že většina koncových uživatelů má na svém počítači jen jeden webový prohlížeč. Kromě toho většina uživatelů nezkoumá vzhled elementů. Běžně si nevšimnou okraje rozevíracího seznamu, který neodpovídá ostatním polím formuláře na dané stránce. Koncoví uživatelé se soustředí na to, aby co nejrychleji z vaší stránky získali požadované informace.

Následuje několik rad, kdy udělat čáru za snahou o stejný vzhled prvků u různých prohlížečů pomocí jednoho stylu jazyka CSS:

- Určité designové prvky byste měli ze stránky vyloučit, pokud úsilí nezbytné k jejich implementaci neodpovídá výsledku.
- Určité designové prvky byste měli ze stránky vyloučit, pokud jejich údržba vyžaduje zásahy do jádra zdrojového kódu, protože podpora takového rozvržení by byla neúnosná.

## Standardy konsorcia W3C pro jazyk CSS

V roce 1996 konsorcium W3C doporučovalo vývojářům standard CSS 1.0. Tento standard podporuje téměř 99 % dnes dostupných prohlížečů.

Aktuální doporučení je CSS 2.1 a přestože není obecně podporováno ve všech prohlížečích, částečnou podporu najdete také v 99 % prohlížečů. Prohlížeče podporují CSS pozicování (před vydáním doporučení CSS 2.1 označované jako CSS-P), kterým můžete umístit elementy na stránce absolutně vzhledem k levému hornímu rohu okna prohlížeče nebo relativně vzhledem k rodičovskému elementu.

Právě se připravuje podpora doporučení CSS 3. Nejlepší podporu CSS 3 má zatím Safari 3, ale poslední verze prohlížečů Firefox a Opera mají také stále lepší podporu. Nebojte se používat pravidla standardů CSS 2.1 a CSS 3, pokud otestujete, že vaše webové stránky vypadají stejně v co největším počtu prohlížečů.

## Pravidla pro šablony stylů

Pokud se budete řídit praxí osvědčenými postupy pro jazyk CSS, pak můžete dosáhnout plně přístupných webových stránek a snadné údržby zdrojového kódu. Pravidla z této kapitoly vám pomůžou připravit šablony stylů na budoucí změny a splnit požadavky koncových zákazníků.

### Oddělení společných pravidel

Je žádoucí, aby každá stránka používala jen ta pravidla jazyka CSS, která potřebuje ke svému zobrazení. Pokud má stránka pravidla, která nepotřebuje, musí prohlížeč tato data stáhnout a interpretovat, než zjistí, že jsou k ničemu. Nesmíte však zapomínat na dvě věci – udržovatelnost a způsob, jakým prohlížeče pracují.

Představte si webové stránky s aktuálními sportovními výsledky skládající se ze čtyř stránek. Domovská stránka popisuje účel webových stránek návštěvníkům a nabízí odkazy na ostatní stránky. Mezi další stránky patří stránka se sportovními výsledky, stránka s novinkami a stránka s často kladenými otázkami (FAQ). Podle tohoto pravidla by jedinými pravidly na dané stránce měla být pravidla nutná k zobrazení této stránky.

Nyní popřemýšlejte, jak by měly tyto stránky vypadat na obrazovce. Spousta webových stránek má komponenty, které se zobrazují na každé stránce, například záhlaví s logem a navigací a zápatí s copyrightem. Většina designů webových stránek je založena na nějaké šabloně rozvržení. Každé stránka by měla mít podobný základ a rozvržení definované designérem. Pomyslete si, jak náročná by byla údržba, kdybyste museli aktualizovat záhlaví, zápatí a rozvržení stránky v případě, že byste měli čtyři soubory s šablonami stylů pro čtyři stránky. Rozumnější přístup musí zajistit udržitelnost a vyhnout se duplikování pravidel stylů.

Všechny hlavní webové prohlížeče se snaží koncovému uživateli stránku zprostředkovat co nejrychleji a používají k tomu spoustu metod. Jednou z nich je *ukládání do mezipaměti*. Tato metoda ukládá kopie dokumentu HTML, obrázků, šablon stylů, souborů jazyka JavaScript a dalších statických prostředků, nutných ke zobrazení stránky, na pevný disk koncového uživatele. To znamená, že když uživatel klepne ve webovém prohlížeči na tlačítko *Zpět* nebo znovu navštíví danou stránku, prohlížeči stačí načíst data z pevného disku uživatele, aby stránku zobrazil. Prohlížeč nemusí stahovat tyto prostředky znovu přes poměrně pomalé internetové připojení, a proto stránku zobrazí mnohem rychleji.

V jednoduchých případech prohlížeč stáhne obrázek, například logo, jedenkrát, když uživatel navštíví domovskou stránku. Za předpokladu, že každá stránka odkazuje na stejný soubor s obrázkem na webovém serveru a obrázek nebyl mezitím aktualizován, webový prohlížeč jej načte z pevného

disku uživatele a neztrácí čas s jeho opětovným stahováním. Stejný princip můžeme aplikovat i na šablony stylů.

Řekněme, že máte na webových stránkách šablonu stylů, která obsahuje pravidla jen pro obecné rozvržení stránky a společné komponenty všech stránek. Na všech stránkách se odkážete na stejný soubor s šablonou stylů. Webový prohlížeč stáhne tento soubor jednou, když uživatel poprvé navštíví libovolnou stránku (obvykle domovskou stánku) a u každé další návštěvy webový prohlížeč použije tento stažený soubor. To urychlí zobrazení stránek, protože se neplýtvá časem na stahování souborů, které nejsou nutné. To je další výhoda toho, když umístíte společná pravidla pro celé webové stránky do jednoho souboru, abyste neměli duplicitní pravidla na svých stránkách.

Specifická pravidla pro jednotlivé stránky můžete umístit do vlastních souborů, takže každá stránka bude odkazovat na dvě externí šablony stylů – jedna společný a jedna unikátní pro danou stránku.

#### Mějte na paměti kaskádové chování a specifičnost

U šablon stylů se setkáte se dvěma důležitými termíny – *kaskádové chování* a *specifičnost*. Kaskádové chování (písmeno C v akronymu CSS) označuje způsob, jakým jsou aplikovány styly aditivně. Specifičnost definuje úrovně, na kterých je příslušný styl aplikován na danou stránku.

Nejlepší způsob, jak psát pravidla, je začít obecnými pravidly a zvyšovat jejich specifičnost, jak pokračujete ve vývoji. Obvykle čím obecnější jsou pravidla, tím efektivnější může být šablona stylů. Snažte se vyhnout příliš specifickým selektorům. Není nutné uvést každý element, identifikátor a třídu, kterými označíte příslušný element – pište stručně a jednoduše.

Podívejte se například na následující obsah elementu `body`:

```
<body>
  <div id="page">
    <div id="header">
      <h1>Titulek mé stránky</h1>
    </div>
    ...
  </div>
<body>
```

Místo následujícího pravidla aplikovaného na element záhlaví:

```
body #page #header h1 {
}
```

použijte o něco jednodušší:

```
#header h1 {
}
```

Jelikož víte, že všechny hodnoty atributu `id` musí být v dokumentu HTML jedinečné, na stránce může být jen jeden element s hodnotou `header` v atributu `id`, takže když uvedete i jeho rodičovský element, přidáváte do pravidla nadbytečnou informaci. Do selektoru nemusíte vkládat element `body` vůbec, protože pravidla smíte aplikovat jen na elementy uvnitř elementu `body`. Předchozí druhé pravidlo rovněž zaručuje, že element s identifikátorem `header` bude samostatnou komponentou nezávislou na svém rodičovském elementu. Jestliže blok `<div id="header">` přesunete na jiné místo zdrojového kódu, do jiného rodičovského elementu, máte jistotu, že pravidlo stylu bude aplikováno stejným způsobem.

Zamyslete se nad dalším příkladem. Předpokládejme, že budete chtít, aby dva odstavce používaly stejné písmo, ale různou velikost a barvu. Zde je příslušný kód jazyka HTML:

```
<body>
  <div id="page">
    <p>Příliš žluťoučký kůň úpěl ďábelské ódy.</p>
    <div class="alternative">
      <p>Ó, náhlý déšť teď zvířil prach a čilá laň běží s houfcem gazel k úkrytům.</p>
    </div>
  </div>
</body>
```

Nyní si prohlédněte následující pravidla:

```
#page p {
  font-family: Verdana, Arial, Helvetica, sans-serif;
  font-size: 1.3em;
  color: #000;
}

#page div.alternative p {
  color: #f00;
  font-size: 1em;
}
```

První obecnější pravidlo bude aplikováno na všechny elementy odstavců uvnitř bloku `<div id="page">`. Druhé pravidlo se aplikuje na specifičtější elementy odstavců uvnitř bloku `<div class="alternative">`. Toto aditivní chování se nazývá *kaskáda*.

Také je zajímavé klíčové slovo `!important`, které lze přidat na konec libovolné vlastnosti uvnitř pravidla a které říká, že dané pravidlo má přednost před jakýmkoliv jiným pravidlem na specifičtější úrovni. Toto klíčové slovo můžete přidat k předchozímu kódu:

```
#page p {
  font-family: Verdana, Arial, Helvetova, sans-serif;
  font-size: 1.3em;
  color: #000!important;
}

#page div.alternative p {
  color: #f00;
  font-size: 1em;
}
```

Protože barva uvnitř obecnějšího pravidla byla označena jako `!important`, bude mít přednost před specifičtějším pravidlem. V tomto případě se barva #000 aplikuje na text všech odstavců ukázkové stránky.

Ve spoustě případů je použití klíčového slova `!important` na vynucení změny priority špatné a ukazuje na špatně navržený stylový předpis. Pokud se kvůli tomu budete muset později k takovému souboru s šablonou stylů vrátit, abyste zjistili, proč některá vámi definovaná pravidla nefungují, nestojí to za ten čas a námahu. Jestliže zjistíte, že musíte používat klíčové slovo `!important`, měli byste zvážit, zda byste neměli šablonu stylů přepsat, aby se pravidla stylů aplikovala na správné elementy. V některých případech možná budete muset přidat nějaký kód HTML ke stránce, aby se styly aplikovaly správně. To je ale stále lepší varianta, než mít špatně navrženou šablonu stylů se zmatenými pravidly.

#### Nezapomeňte na tiskárnu

Většina vývojářů zapomíná při psaní šablon stylů na tiskárnu, ačkoliv jen zřídka je toto opomenutí záměrné. Je důležité, abyste věděli, co se vytiskne, a co ne, když uživatel klepne na tlačítko pro tisk na nástrojové liště webového prohlížeče. Nesmíte zapomenout na tři věci:

- Široký obsah: obsah přesahující šířku stránky je obvykle oříznut a nevytiskne se. Některé prohlížeče umí zmenšit obsah tak, aby se vlezl na stránku, ale pak může být text špatně čitelný.
- Obrázky a barva pozadí: protože jako obrázky v popředí byste měli vkládat jen obrázky, které přímo souvisí s obsahem stránky, všechny ostatní obrázky (například loga) by měly být obrázky pozadí. Většinou všechny obrázky budou obrázky pozadí, definované externími soubory se šablonami stylů jazyka CSS. Většina prohlížečů nevytiskne tyto obrázky a barvy pozadí. Výrobci webových prohlížečů udělali chytrý tah a uživatelům ušetřili inkoust tak, že vytisknou jen to, co je potřeba – textový obsah stránky. Jak bylo řečeno dříve, uživatelé většinou tisknou webovou stránku, aby si přečetli její hlavní obsah – ať už jde o vlakový jízdní řád, recenzi knihy nebo bankovní výpis. Návštěvník stránek chce jen výjimečně vytisknout stránku přesně tak, jak ji vidí na obrazovce. Pro ty, kteří chtějí stránku vytisknout kompletní, nabízí prohlížeče nastavení, která povolí tisk obrázků a barev pozadí, ale tato nastavení jsou jen vzácně ve výchozím nastavení povolená.
- Barva textu: většina uživatelů má v kancelářském prostředí přístup jen k laserové tiskárně nastavené na černobílý tisk. Většina domácích uživatelů má barevnou inkoustovou tiskárnu, která umí tisknout ve fotografické kvalitě. Předpokládejte, že uživatel nemusí mít barevnou tiskárnu a zkuste vytisknout své webové stránky na černobílé tiskárně, máte-li nějakou, nebo nastavte svou barevnou tiskárnu na černobílý tisk. Barva textu hraje důležitou roli v kombinaci s pozadím. Představte si, že vaše stránka má bílý text na tmavém pozadí. Protože barva pozadí se nevytiskne, zůstane vám jen bílý text na bílém papíře. Tiskárna samozřejmě ve skutečnosti bílý inkoust nemá, takže uživatel vytiskne prázdnou stránku.

Tiskový výstup webové stránky obvykle obsahuje prázdná místa tam, kde původně byly obrázky pozadí, text přetékající přes okraj stránky a neviditelný světlý text. Tohle chtějí uživatelé jen stěží vytisknout.

Jelikož nechcete, aby vaši uživatelé byli zmatení, použijete šablonu stylů, která bude sloužit jen pro tisk, a to pomocí následujícího elementu:

```
<link rel="stylesheet" href="master-print.css" media="print" />
```

V šabloně stylů pro tisk budete chtít minimalizovat počet bílých znaků, odstranit nepotřebné komponenty a zajistit, aby žádný obsah nebyl na vytisknuté stránce uříznutý nebo schovaný. Například navigaci je možné odstranit následujícím pravidlem:

```
#navigation {
  display: none;
}
```

Možná budete chtít specifikovat černou barvu písma, která přebije všechny ostatní barvy písma na stránce. Toho lze dosáhnout následujícím pravidlem:

```
* {
  color: #000!important;
}
```

Toto pravidlo aplikuje černou barvu písma (černá se zapisuje hexadecimálně jako #000000, zkráceně #000) na všechny elementy dané stránky (zástupný znak * představuje všechny elementy stránky). Všimněte si klíčového slova `!important` na konci vlastnosti. Jak již bylo řečeno, toto klíčové slovo může být velmi nebezpečné, protože upřednostňuje vlastnost uvnitř daného pravidla nad stejnou vlastností uvnitř jakéhokoliv specifičtějšího pravidla. V šabloně stylů pro tisk je však velmi užitečné, protože umožní předefinovat barvu písma jakéhokoliv elementu na stránce, kterou dříve definovala hlavní šablona stylů, a to aniž byste museli přemýšlet nad úrovněmi specifičnosti.

#### Formátujte svá pravidla

Následující příklad ukazuje ověřený formát zápisu pravidel, který je velmi přehledný:

```
.module-heading,
.module-subheading {
  background: #333;
  color: #f00;
  float: left; /* řádkový komentář */
}
```

Předchozí pravidlo má dva selektory, díky kterým bude aplikováno na elementy se dvěma různými třídami. Je lepší zapsat každý selektor na samostatný řádek, protože na jednom řádku by mohly působit nepřehledně. Pravidlo začíná složenou závorkou na konci řádku s názvem poslední třídy a končí na samostatném řádků za deklaracemi všech jeho vlastností. Samotné deklarace vlastností jsou odsazené jedním tabulátorem vzhledem k danému pravidlu. Mezi vlastností a její hodnotou je dvojtečka následovaná mezerou.

Každá deklarace vlastnosti je umístěna na samostatném řádku a každý řádek končí středníkem. Pokud potřebujete k deklaraci vlastnosti přidat komentář, který by měl pomoct s budoucí údržbou, měli byste jej vložit za středník na řádek s danou deklarací vlastnosti. Lepší konzistence pravidel můžete dosáhnout abecedním uspořádáním deklarací vlastností podle názvů vlastností, podle typu vlastnosti (písmo, barva, pozice atd.) nebo jiným způsobem, který nejlépe vyhovuje vaší aplikaci.

#### Přiřaďte více tříd jedinému elementu

Jak bylo řečeno dříve, názvy tříd jazyka CSS by měly popisovat, co daný element představuje, a ne jak má být zobrazen.

Možná budete chtít přiřadit jednomu elementu několik tříd, abyste nemuseli duplikovat pravidla. Toho lze dosáhnout tak, že jednotlivé třídy jazyka CSS v HTML kódu oddělíte mezerou, jak ukazuje tento příklad:

```
<div class="article main-article">
  <p>Ahoj světe.</p>
</div>
```

Na daný element `div` se uplatní pravidla pro třídy `article` a `main-article`, zleva doprava, takže definice vlastností v pravidlu třídy `main-article` mají přednost před definicemi vlastností uvnitř pravidla třídy `article`.

Pokud chcete aplikovat speciální pravidlo na všechny elementy, které mají několik tříd, použijete zápis selektoru jako u následujícího pravidla:

```
div.article.main-article p {
  color: #0f0;
}
```

Toto pravidlo se uplatní u všech elementů `p`, jejichž rodičovským blokem je `<div class="article main-article">`. Všimněte si chybějící mezery mezi názvy tříd v selektoru.

#### Resetujte výchozí styly prohlížeče

Webový prohlížeč má standardně sadu výchozích pravidel, která se aplikují na stránku bez vlastních pravidel. Tato pravidla definují písmo, jeho velikost, výšku řádku, barvu, vnitřní a vnější okraj pro různé elementy.

Abyste získali slušný základ pro své vlastní styly, doporučuji resetovat výchozí pravidla prohlížeče. Díky tomu získáte lepší kontrolu nad každým elementem ve svých šablonách stylů a také lepší konzistenci mezi různými prohlížeči.

Například následující pravidlo nastaví vnitřní a vnější okraj všech elementů stránky na nulu, což vám umožní specifikovat vlastní vnitřní a vnější okraje pro každý element:

```
* {
  margin: 0;
  padding: 0;
}
```

Eric Meyer, guru jazyka CSS, nabízí komplexní resetovací šablonu stylů, která zcela odstraní rozdíly mezi výchozími pravidly různých prohlížečů (*http://meyerweb.com/eric/tools/css/reset/*). Aktuální verzi této šablony vidíte ve výpise 1.1.

**Výpis 1.1.** Univerzální resetovací šablona stylů od Erica Meyera.

```
/* v1.0 | 20080212 */
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, font, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td {
    margin: 0;
    padding: 0;
    border: 0;
    outline: 0;
    font-size: 100%;
    vertical-align: baseline;
    background: transparent;
}

body {
    line-height: 1;
}

ol, ul {
    list-style: none;
}
```

```
blockquote, q {
    quotes: none;
}

blockquote:before, blockquote:after,
q:before, q:after {
    content: '';
    content: none;
}

/* nezapomeňte definovat pravidla zaměření vstupu od uživatele */
:focus {
    outline: 0;
}

/* nezapomeňte zvýraznit vložené texty */
ins {
    text-decoration: none;
}

del {
    text-decoration:  line-through;
}

/* tabulky v dokumentu musí mít nulové odsazení buněk */
table {
    border-collapse: collapse;
    border-spacing: 0;
}
```

Eric Meyer strávil spoustu času zkoumáním podpory jazyka CSS ve všech hlavních prohlížečích a jeho resetovací šablona je třešničkou na dortu této práce.

Bez ohledu na to, zda použijete pravidlo pro nastavení nulových vnitřních a vnější okrajů, resetovací šablonu od Erica Meyera z výpisu 1.1, nebo nějakou z dalších alternativ, kterou můžete najít pří zběžném hledání na webu, měli byste tato resetovací pravidla definovat na začátku hlavního souboru s šablonou stylů, tedy ještě předtím, než definujete další pravidla.

#### Hlavní zkrácená pravidla

Určité definice vlastností lze kombinovat a zkrátit. Zkrácené verze byste měli používat, kde je to možné, protože tím snižujete velikost souborů s šablonami stylů a také objem dat přenášených z webového serveru do prohlížeče.

Například hodnoty vlastností `margin` a `padding` je možné definovat dlouze:

```
p {
  margin-top: 5px;
  margin-right: 7px;
  margin-bottom: 5px;
  margin-left: 7px;
  padding-top: 6px;
  padding-right: 3px;
  padding-bottom: 6px;
  padding-left: 3px;
}
```

nebo zkráceně:

```
p {
  margin: 5px 7px 5px 7px;
  padding: 6px 3px 6px 3px;
}
```

Všimněte si, že pořadí zkrácených hodnot vlastností `margin`, `padding` a `border` vždy odpovídá tomuto:

- nahoře,
- vpravo,
- dole,
- vlevo.

Když se shodují hodnoty vlastností `margin-top` s `margin-bottom` a hodnoty vlastností `margin-left` s `margin-right`, lze použít podobný zápis jako u následujícího pravidla:

```
p {
  margin: 5px 7px; /* 5 px  nahoře a dole, 7 px vlevo a vpravo*/
  padding: 6px 3px;
}
```

Další vlastnosti stylů, například `font`, `color` a `background`, mají také svůj zkrácený zápis. Kapitola 4, která se zaměřuje na efektivitu souborů s šablonami stylů, obsahuje více informací o zkrácených zápisech definic vlastností.

## Pravidla přístupnosti pro styly

Jedním z hlavních cílů vývojáře webových aplikací je přístupnost – zajištění toho, aby byl obsah dostupný pro každého, bez ohledu na to, jaký má webový prohlížeč nebo zařízení. Zejména se klade důraz na sémantiku zdrojového kódu, ale v záloze musíte mít i šablony stylů. Uživatelé s různým rozlišením a velikostí obrazovky by měli vidět obsah webové stránky správně.

#### Schovávejte obsah v prohlížečích s podporou jazyka CSS

Pomocí šablon stylů můžete schovat obsah některých elementů v dokumentu HTML, pokud je nechcete zobrazit na obrazovku. Vzpomeňte si na příklad z této kapitoly s webovou stránkou s recenzemi filmů, kde jste chtěli umístit text přímo do zdrojového kódu jazyka HTML, abyste měli jistotu, že bude přečten nahlas nebo bude viditelný bez přítomnosti šablony stylů, aby tak uživatel poznal, jaké hodnocení recenzent filmu dal. Nyní je čas vaši šablonu stylů aplikovat. Místo daného textu budete chtít zobrazit obrázek se čtyřmi hvězdičkami z pěti.

Text je možné schovat pomocí vlastnosti `text-indent`, pokud jí přiřadíte hodnotu, která umístí text před levý okraj okna webového prohlížeče, takže už nebude vidět:

```
div#hide-me {
  text-indent: -9999px;
}
```

Jestliže schováte text pomocí jiné vlastnosti, nebude tento text čitelný oblíbenou čtečkou obrazovky s názvem JAWS, kterou vydala společnost Freedom Scientific (*http://www.freedomscientific.com/*).

#### Přesouvejte bloky s obsahem, abyste udrželi správné pořadí ve zdrojovém kódu

Dříve jste se v této kapitole dozvěděli, že bloky s obsahem by měly být v kódu jazyka HTML uspořádané tak, aby v případě, že bude obsah stránky předčítán nahlas, byl důležitý obsah přečten jako první a nepodstatný obsah, včetně navigace a copyrightu, jako poslední. Samozřejmě, že toto uspořádání nemusí odpovídat vzhledu stránky, kde pravděpodobně budete chtít zobrazit navigaci nad hlavním obsahem.

Pokud máte ve zdrojovém kódu dva blokové elementy za sebou a chtěli byste zobrazit druhý element před prvním, můžete tyto elementy přeuspořádat pomocí vlastnosti `float`. Podívejte se na následující ukázku HTML kódu:

```
<div id="body">
  ...
</div>

<div id="navigation">
  ...
</div>
```

Běžně by se blok `<div id="body">` objevil před blokem `<div id="navigation">`, ale toto pořadí lze obrátit vlastností `float` takto:

```
#body {
  float: right;
}

#navigation {
  float: left;
}
```

Mohli byste také zvážit, zda obsahové bloky nerozmístit na stránce kombinací absolutního a relativního pozicování, abyste dosáhli požadovaného rozložení.

#### Používejte relativní velikost písma

Někteří návštěvníci vašich webových stránek, kteří špatně vidí nebo mají velké rozlišení obrazovky, budou chtít pravděpodobně zvětšit velikost písma v prohlížeči. Všechny hlavní webové prohlížeče mají tuto funkci. Jako vývojáři musíte vzít v úvahu kromě designu svých webových stránek také potřeby koncových uživatelů .

Abyste umožnili změnu velikosti písma, měli byste ve svých šablonách stylů používat relativní jednotky. Když se zvětší velikost písma v prohlížeči, měla by se adekvátně zvětšit i velikost písma na dané stránce. Jednou z relativních jednotek je jednotka em, protože je vyjádřena v procentech. Ve svých šablonách stylů byste rozhodně neměli používat velikost písmen s jednotkami px (pixel) nebo pt (bod), protože ty se nepřizpůsobí nastavení velikosti písma webového prohlížeče. Výpočty s relativními velikostmi písma si usnadníte tak, že nastavíte výchozí velikost písma elementu `body` na `62.5%`. Ve výchozím nastavení prohlížeče bude pak hodnota `1em` odpovídat velikosti `10px`. Každé zvýšení této hodnoty o `0.1em` bude srovnatelné se zvýšením o `1px`, proto `1.1em` bude ekvivalentní s `11px` a `1.2em` s `12px`. Tento trik je užitečný zejména tehdy, když pracujete s kreativním designem, který má často velikost písma určenou počtem pixelů. Příklady velikostí:

```
body {
  font-size: 62.5%; /* nastaví velikosti písma tak, že 1em bude vzhledově
    odpovídat 10px */
```

```
}

ul li {
  font-size: 1.1em; /* odpovídá 11px */
}

h2 {
  font-size: 1.5em; /* odpovídá 15px */
}
```

## Komentářové bloky

Přehlednějších šablon stylů dosáhnete seskupením logicky souvisejících stylových pravidel. Před skupinu pravidel vložte komentář, který popisuje účel této skupiny pravidel, a to pokud možno komentář s konzistentní strukturou, například:

```
/* ----------------------------------------------------------------------
    Pravidla formulářů

    Skupina pravidel zobrazujících formulářové prvky
    podle stanoveného designu.
*/
```

Řádkové komentáře byste měli použít, jen když je nutné vysvětlit určité pravidlo. Například pravidlo, které se má vypořádat s rozdílným zobrazením okrajů v prohlížečích, by mělo mít komentář, aby jej v budoucnosti některý vývojář nepokládal za zbytečné a neodstranil jej, protože nepochopil jeho význam.

Na úplném začátku souboru s šablonou stylů by měl být komentář popisující autora souboru, účel obsažených pravidel a seznam logických skupin seřazených podle výskytu v souboru, a to v nějakém jednotném formátu. Zde je příklad zápisu komentářového bloku popisujícího celou šablonu stylů a její účel:

```
/* ----------------------------------------------------------------------
    Název souboru        common.css
    Autor                Den Oddel me@denodell.com
    Popis                Základní rozložení stránky a výchozí pravidla pro webové stránky

    Obsah
    - Vynulování výchozích nastavení prohlížečů
    - Rozložení sloupců stránky
    - Pravidla formulářových prvků

    Barevná paleta
    #777           středně šedá
    #aaa           světle šedá
    #a3d60a        zelená
---------------------------------------------------------------------------
*/
```

Díky tomuto komentářovému bloku vývojář na první pohled pozná, jestli daný soubor obsahuje kód, který hledá. V případě šablon stylů společných pro všechny stránky je dobré vložit do něj i popis klíčových barev vašich webových stránek. Vývojáři pak mohou snadno tyto barvy vyhledat a zkopírovat je nebo nahradit.

## Obcházení nedostatků webových prohlížečů

Pokud pro některé verze prohlížeče IE potřebujete zvláštní pravidla, měli byste pomocí podmínkových komentářů vkládat samostatné soubory s šablonami stylů pro tato pravidla. Tato technika byla vysvětlena dříve v této kapitole.

**Upozornění**

Vyhýbejte se hackům typu vkládání podivných hodnot, které kombinují zpětná lomítka, komentáře a další divné znaky, aby zmátly některé prohlížeče. Pokud není jiná možnost a musíte použít nějaký hack, popište komentářem, proč jste jej vložili a jaký má účel. Pomůžete tak vývojářům, kteří si v budoucnu mohou prohlížet vaši šablonu stylů.

V prohlížeči IE 6 a ve starších verzích se u obrázků ve formátu PNG-24 zobrazují průhledné části obrázků jako šedá/světle modrá barva. Bohužel, společnost Microsoft zavedla podporu průhlednosti těchto obrázků až ve verzi IE 7. Můžete však použít trik pro správné zobrazení obrázků typu PNG-24 v těchto starších prohlížečích.

Řekněme, že vaše šablona stylů obsahuje následující pravidlo:

```
#header {
  background: url(muj-obrazek.png) no-repeat;
}
```

Do své šablony stylů určené pro IE 6 a vložené podmínkovým komentářem byste vložili podobné pravidlo:

```
#{
  background: transparent none;
  filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='muj-obrazek.png',
    sizingMethod='scale');
}
```

Předchozí pravidlo pro prohlížeč IE 6 skryje obrázek pozadí definovaný v původní šabloně stylů a použije filtr knihovny DirectX od společnosti Microsoft s odkazem na stejný soubor s obrázkem. Tento filtr umí zobrazit průhledný obrázek správně a umístí jej na stejné místo jako původní obrázek pozadí. Tato technika má svá úskalí, protože chybí podpora pozicování a opakování obrázků na pozadí, proto ji používejte obezřetně.

## Nezapomeňte na lokalizace

Důležitou věcí při tvorbě webových stránek je myslet na to, že vaše stránky mohou být v budoucnosti přeloženy do dalších jazyků nebo mohou mít regionální verze.

Pravděpodobně se nezmění jen textových obsah webových stránek, ale i nějaká část šablon stylů. Například můžete chtít vyměnit obrázek s textem v určitém jazyce za jiný nebo pro některé abecedy vyměnit směr textu.

Proto byste měli všechna jazykově nezávislá pravidla umístit do hlavních souborů s šablonami stylů a jazykově závislá pravidla do samostatných šablon stylů. Vybraná jazykově závislá šablona stylů pak přebije hlavní šablonu stylů. Tímto způsobem můžete snadno podporovat nové jazyky, protože jen

vytvoříte novou šablonu stylů zkopírováním některé jazykově závislé šablony stylů a změnou některých pravidel podle aktuálního jazyka.

# Uspořádání složek, souborů a prostředků

Jednoduchá rozšiřitelná struktura složek je dobrý základ pro budoucí rozšiřování obsahu a prostředků, aniž by se z toho stala pro vývojáře noční můra. Následující části obsahují rady pro uspořádání vašich složek, souborů a prostředků.

## Čitelné URL adresy

Složky byste měli vytvářet s ohledem na mapu webových stránek vašeho projektu, aby dokumenty HTML a skripty na straně serveru byly uloženy ve složkách tak, že jim odpovídající URL adresy budou čitelné a smysluplné. Obvykle byste měli nakonfigurovat webový server tak, aby se v každé složce zobrazovala výchozí (indexová) stránka automaticky. Díky tomu můžete v adresách URL používat jen odkazy na složky a indexový soubor se zobrazí automaticky.

Měli byste vytvářet strukturu složek v souladu s mapou webových stránek, aby adresy URL také odpovídaly dané struktuře. Některé vyhledávače používají adresy URL a obsah stránek jako kritéria hledání.

Například tato adresa URL:

*http://www.mojewebovestranky.cz/novinky/*

je čitelnější a lépe zapamatovatelná než tato:

*http://www.mojewebovestranky.cz/novinky.php*

Části adresy URL by rovněž měly ctít mapu webových stránek a měly by mít smysluplnou strukturu, například:

*http://www.mojewebovestranky.cz/hudba/rock/*

## Pojmenování souborů a složek

Při pojmenovávání složek a souborů byste měli používat malá písmena a pro přehlednost každé slovo oddělit pomlčkou. Vyhněte se jiným znakům než písmenům bez diakritických znamének, číslům a pomlčkám. Když se na tyto soubory a složky odkazujete ve zdrojovém kódu, ujistěte se, že vždy používáte malá písmena, protože některé operační systémy, například Unix, Linux a Mac OS X rozlišují velikost písmen. Nikdy v názvu souboru nebo složky nepoužívejte mezeru a písmena vyhrazená pro adresy URL, včetně ampersandu, mřížky a otazníku, jinak zjistíte, že v některých webových prohlížečích se na takové soubory neodkážete.

Jedním z požadavků, který se neustále vynořuje, je poskytnout koncovým uživatelům obsah lokalizovaný do jejich mateřského jazyka. Tato lokalizace může být neuvěřitelně složitá, pokud ji začnete řešit až po dokončení webových stránek, protože by pravděpodobně vyžadovala přepsání některých souborů jazyka HTML, CSS a JavaScript. Prostředky byste měli seskupovat dle jejich jazyka do vlastních složek pojmenovaných tímto jazykem (například zkratkou `cs` pro češtinu). Jazykově nezávislé prostředky byste měli uložit na stejnou úroveň, jako je tato struktura složek.

**Tip**

Rozmyslete se, jestli nepoužijete soubory XML, server pro správu obsahu nebo soubory na straně serveru pro přímé generování lokalizované verze webových stránek. Tento typ údržby je méně náročný než upravovat statické dokumenty HTML.

## Znaková sada souborů

U všech textových souborů byste měli používat znakovou sadu UTF-8, díky tomu budete moct ve svých souborech používat speciální znaky a písmena české abecedy, aniž byste museli používat speciální řídící znaky nebo kódy ASCII.

Je jednodušší vytvářet soubory přímo ve znakové sadě UTF-8, než je později na tuto znakovou sadu převádět. Jestliže potřebujete převést soubor z jiné znakové sady na UTF-8, zálohujte si stávající soubory a pomocí dialogového okna s vlastnostmi souboru ve vašem vývojovém prostředí změňte znakovou sadu. To může změnit některé znaky ve vašem souboru, proto si otevřete soubor zálohy a zkopírujte z něj tyto znaky do souboru se znakovou sadou UTF-8.

V některých případech může značka BOM (byte-order mark) na začátku souboru se znakovou sadou UTF-8 způsobit problémy v technologii na straně serveru, například v jazyce PHP. Pokud převádíte soubory, které mají být zpracovány těmito technologiemi, ověřte si v dialogovém okně s vlastnostmi souboru ve vašem vývojovém prostředí, že značka BOM bude odstraněna (pokud tam takové nastavení je).

## Uspořádání prostředků

Zvažte, zda neumístíte všechny soubory prostředků (šablony stylů, obrázky, skripty, zvukové soubory a videosoubory), kromě dokumentů HTML, do společné složky. To logicky oddělí zdrojový kód na straně klienta od struktury a obsahu webových stránek, nebudete tak mít nepořádek v samotném kořenovém adresáři dokumentů. Následuje příklad struktury složek, která se řídí tímto přístupem a nezapomíná na lokalizaci do cizích jazyků:

```
\prostredky
  \obrazky
    \cs-cs
    \en-gb
    \fr-fr
  \skripty
    \treti-strany
  \styly
    \cs-cs
    \en-gb
    \fr-fr
  \flash
  \dokumenty
    \cs-cs
    \en-gb
    \fr-fr
  \video
    \cs-cs
    \en-gb
    \fr-fr
```