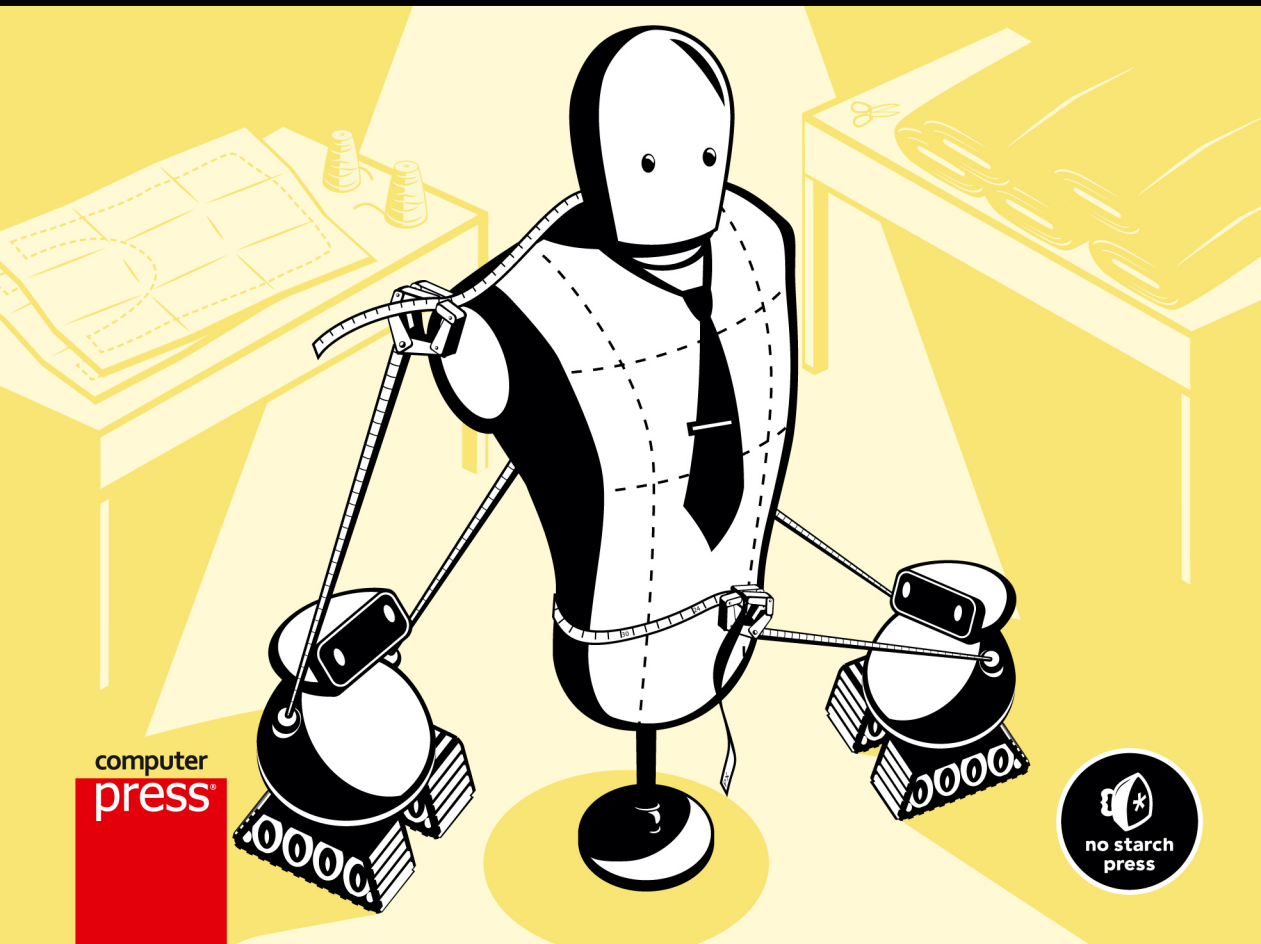
CSS3

Průvodce vývojáře moderního webdesignu

PETER GASSTON



computer press

no starch press

Peter Gasston

CSS3

Computer Press
Brno
2016

CSS3

Peter Gasston

Překlad: Ondřej Baše
Ilustrace na obálce: Octopod Studios a Garry Booth
Odpovědný redaktor: Martin Herodek
Technický redaktor: Jiří Matoušek

Objednávky knih:
http://knihy.cpress.cz
www.albatrosmedia.cz
eshop@albatrosmedia.cz
bezplatná linka 800 555 513
ISBN 978-80-251-4641-5

Vydalo nakladatelství Computer Press v Brně roku 2016 ve společnosti Albatros Media a. s. se sídlem Na Pankráci 30, Praha 4. Číslo publikace 23 612.

1. vydání

ALBATROS MEDIA a.s.

Obsah

Pro mou sestru Sáru. Tvá odvaha mě inspirovala.

Předmluva

Tato kniha je výsledkem osmi let psaní o jazyce CSS3, a to jak na webu, tak v tištěné podobě. Jazyk CSS a prohlížeče prošly za tak krátkou dobu řadou změn. Neustále se mění a vznikají nové funkce frekvencí, s jakou lze jen těžko držet krok. Specifikace CSS3 je napsána velmi technickým stylem a je určena spíše výrobcům prohlížečů než koncovým uživatelům. Účelem této knihy je překlenout mezeru mezi příliš technickou specifikací a běžným webovým vývojářem.

Tato kniha je volně uspořádaná podle stability implementace. V úvodních kapitolách se nacházejí vlastnosti jazyka CSS s širokou podporou, které vývojáři používají každodenně, a přitom postupně přejdeme k experimentálnějším funkcím, s nimiž se lze setkat v užším okruhu prohlížečů. V závěrečných kapitolách se kniha mnohdy spoléhá jen na výklad specifikace CSS3, aby popsala, jak by se měly chovat některé budoucí vlastnosti.

Doufám, že jsem při psaní této knihy udělal jen minimum chyb, ale pokud na nějaké narazíte, nejspíše budou důsledkem toho, že jsem špatně pochopil tuto specifikaci.

Při psaní této knihy jsem čerpal z nejrůznějších modulů specifikace CSS3 a také ze stránek Mozilla Developer Network (*https://developer.mozilla.org/*) – jedinečné sbírky článků o všem, co se děje na webu (včetně dění okolo jazyka CSS), které jsou o to zajímavější, že je píšou dobrovolníci.

Spousta ukázkových zdrojových kódů používá texty z volně dostupných knih. U obrázků v knize, které jsem nevytvořil osobně, jsem vyjádřil své poděkování jejich autorům v jednotlivých kapitolách.

Tato kniha by nevznikla nebýt pomoci týmu nakladatelství No Starch Press, a to zejména redaktorky Sereny Yang a korektorů Keitha Fanchera (první vydání) a Billa Pollocka (druhé vydání). Díky nim jsem se naučil psát čistě a z blogera se ze mě stal autor. Rád bych také poděkoval odborným korektorům: Patricku Laukemu – jeho smysl pro detail a znalost technických specifikací pro mě byly naprosto klíčové při tvorbě druhého vydání; a Joostovi de Valk, který byl nejen mým odborným korektorem pro první vydání, ale také mě inspiroval, abych psal o jazyce CSS3, když před osmi lety vytvořil webové stránky *http://www.css3.info/*.

Rovněž bych chtěl poděkovat svým kolegům na projektech Preloaded, Poke, Top10 a Rehabstudio za to, že mě podporovali a povzbuzovali k napsání dvou vydání této knihy. Děkuji všem lidem na nesčetných setkáních webové komunity v Londýně, své mámě, že mě naučila, jakou hodnotu má těžká práce, a tátovi za to, že mi před téměř třiceti lety koupil první počítač. Slibuji, že mu ho jednoho dne splatím; tato kniha mi snad s tímto dluhem pomůže.

Úvod

Nechte mě, abych vám pověděl, co si myslím – myslím si, že jste webový profesionál, který píše kód v jazycích HTML a CSS již roky, umíte rozlišit nejen element `div` od elementu `span`, ale také element `b` od elementu `strong`. Máte určité povědomí o jazyce CSS3 a začal jste experimentovat s některými jeho dekorativními vlastnostmi – například s kulatými rohy, ale chcete lépe pochopit tento jazyk.

Tato kniha vám pomůže zužitkovat vaše úžasné znalosti o jazyce CSS2.1, abyste se mohli snadněji naučit jazyk CSS3. Nebudu zde popisovat základy jazyka CSS (až na příležitostné poznámky), protože předpokládám, že už je znáte. Nebudu zde podrobně vysvětlovat, jak používat jazyk CSS k tvorbě navigace nebo obrázkové galerie, jelikož z příkladů v této knize budete moct vytvořit cokoliv sami.

Představím vám, co můžete dělat s jazykem CSS3 dnes a co budete moct dělat v budoucnosti. Přitom vezmu technický jazyk specifikace CSS3 a přeložím ho do prostého jazyka, který je praktičtější.

Snad zde najdete nové nástroje pro vývoj, s nimiž budete moct dělat ještě úžasnější věci.

Co najdete v této knize

Jazyk CSS je možné používat napříč různými typy médií – téměř všechna zařízení, která umějí pracovat s jazyky HTML a XML, umí pracovat také s pravidly stylů, třebaže mnohdy jen v omezené podobě. Jazyk CSS3 obsahuje moduly určené výhradně pro stránkovaná média, jako jsou dokumenty PDF nebo tištěné materiály. Podporuje Braillovo písmo, příruční mobilní zařízení (chytré telefony), teletypy a televize. Rozsah možností je tak rozsáhlý, že je nemůžu pokrýt všechny.

Tato kniha se zaměřuje na použití jazyka CSS pro počítačové obrazovky. Všechny ukázky byly napsány (a otestovány) pro nejrozšířenější stolní verze prohlížečů a optimalizovány pro uživatele stolních počítačů a notebooků. Většina nových vlastností jazyka CSS popsaných v této knize by měla fungovat, ať už vyvíjíte pro chytré telefony, tablety nebo jiná zařízení, ale nemůžu zaručit, že všechno budete vypadat přesně jako ve zde uvedených příkladech.

Kapitola za kapitolou

Následuje stručný souhrn toho, co obsahuje tato kniha:

- **Kapitola 1** představuje specifikaci CSS3, její historii a standardizační proces konsorcia W3C. Popisuje rovněž syntaxi použitou v ukázkových zdrojových kódech.
- **Kapitola 2** pokrývá dotazy na médium a technologie pro adaptivní a responzivní webdesign.
- **Kapitoly 3 a 4** představují nové selektory – selektory atributů v kapitole 3 a pseudotřídy v kapitole 4.
- **Kapitola 5** ukazuje, jak vybrat vlastní webová písma, a také vlastnosti, s nimiž získáte lepší kontrolu nad zobrazováním písma.
- **Kapitola 6** navazuje na téma typografie. Věnuje se vlastnostem pro doplňování stínů k textu a řízení způsobu zobrazování textových bloků.
- **Kapitola 7** je také o textu – vysvětluje, jak řídit tok textu napříč více sloupci.
- **Kapitoly 8 a 9** popisují moduly Background a Borders, a to včetně rozšíření stávajících vlastností pozadí a zcela nových dekorativních efektů pro rámečky.
- **Kapitola 10** vysvětluje, jak pracovat s průhledností, a představuje několik nových metod pro definici barev.
- **Kapitola 11** představuje barevné přechody, což jsou pozvolné přechody mezi dvěma a více barvami, pomocí nichž lze dosáhnout jedinečných efektů na pozadí.
- **Kapitoly 12 a 13** ukazují, jak vizuálně měnit elementy ve dvou a třech rozměrech.
- **Kapitola 14** představuje animace; například postupné změny mezi dvěma hodnotami a složité časované animace.
- **Kapitola 15** popisuje Flexbox, což je nová metoda pro rozvrhování elementů na základě dostupného prostoru.
- **Kapitola 16** se rovněž zabývá rozvrhováním stránky, představuje nové jednotky a vysvětluje, jak počítat rozměry a měnit velikost elementů podle jejich obsahu.
- **Kapitola 17** je poslední kapitola o rozvrhování, která představuje nový modul Grid Layout jazyka CSS.
- **Kapitola 18** se zaměřuje na vizuální efekty; například na míchání vrstev pozadí nebo jednoho elementu s druhým, používání grafických filtrů a na to, jak ořezávat elementy pomocí jednoduchých tvarů.
- **Kapitola 19** uzavírá tuto knihu a nastiňuje možnou budoucnost jazyka CSS – představuje nové vlastnosti, které jsou zatím hodně experimentální, ale v budoucnu se můžou stát běžnou součástí webových prohlížečů.

Přílohy a další zdroje

Na konci této knihy najdete dvě přílohy. První z nich obsahuje stručný přehled podpory vlastností jazyka CSS popisovaných v této knize napříč webovými prohlížeči. Druhá obsahuje seznam online zdrojů, užitečných nástrojů a zajímavých příkladů.

Kromě těchto doprovodných webových stránek můžete navštívit také můj blog Broken Links (*http://www.broken-links.com/*), na němž najdete další články o jazyce CSS3 (a jiných nových webových technologiích). Můžete mi zde napsat i nějaký komentář nebo mě kontaktovat.

Úvod do druhého vydání

Na prvním vydání této knihy jsem začal pracovat v roce 2010. Je to jen pět let, ale jak moc se vše změnilo za tu dobu! V roce 2010 byl iPad na světě jen několik měsíců, připravoval se operační systém Android, a když jsem sledoval statistiky návštěvnosti svých webových stránek, zaznamenal jsem 11,6procentní návštěvnost z mobilních zařízení – v době psaní této knihy už se ale jednalo o 54,8procentní podíl.

Od prvního vydání této knihy vyšly čtyři hlavní verze prohlížeče Safari, tři verze prohlížeče Internet Explorer, prohlížeč Firefox přešel na skryté automatické aktualizace, jádro prohlížeče Chrome, WebKit, zavedlo vlastní rozvrhovací jádro Blink, Opera se vzdala práce na vlastním jádru Presto a místo toho také používá WebKit.

Navíc vzestup preprocesorů Sass a LESS přinesl sílu programovacích jazyků do šablon stylů a razantně změnil způsob, jakým píšeme kaskádové styly. Většina webových vývojářů nyní používá nějaký preprocesor jazyka CSS jako základní komponentu pro vývoj webových stránek.

Spousta specifikací jazyka CSS se rovněž změnila od prvního vydání. Některé byly zrušeny a jiné vznikly. S jazykem CSS3 bylo v roce 2010 spojeno mnohem více implementačních rozdílů mezi prohlížeči a dnešní rozdíly jsou mnohem drobnější, jelikož výrobci prohlížečů se více spoléhají na standardy.

Jinými slovy – druhé vydání není pouze lehkou úpravou prvního vydání. Všechny kapitoly byly plně revidovány, aby pokryly všechny změny specifikace, byly z nich odstraněny zastaralé informace a experimentální vlastnosti, které již nejsou součástí specifikace. Některé kapitoly (zejména o dotazech na médium, modulu Flexbox, mřížkách a budoucnosti jazyka CSS) jsou téměř zcela nové. Kromě toho přibyly nové kapitoly o hodnotách a rozměrech a také o režimech míchání, efektech filtrů a maskování.

To by snad mělo stačit na dalších pět let změn.

Zpětná vazba od čtenářů

Nakladatelství a vydavatelství Computer Press, které pro vás tuto knihu přeložilo, stojí o zpětnou vazbu a bude na vaše podněty a dotazy reagovat. Můžete se obrátit na následující adresy:

Computer Press
Albatros Media a.s., pobočka Brno
IBC
Příkop 4
602 00 Brno

nebo

sefredaktor.pc@albatrosmedia.cz

Computer Press neposkytuje rady ani jakýkoli servis pro aplikace třetích stran. Pokud budete mít dotaz k programu, obraťte se prosím na jeho tvůrce.

Zdrojové kódy ke knize

Z adresy *http://knihy.cpress.cz/K2223* si po klepnutí na odkaz Soubory ke stažení můžete přímo stáhnout archiv s ukázkovými kódy.

Errata

Přestože jsme udělali maximum pro to, abychom zajistili přesnost a správnost obsahu, chybám se úplně vyhnout nelze. Pokud v některé z našich knih najdete chybu, ať už chybu v textu nebo v kódu, budeme rádi, pokud nám ji oznámíte. Ostatní uživatele tak můžete ušetřit frustrace a pomoci nám zlepšit následující vydání této knihy.

Veškerá existující errata zobrazíte na adrese *http://knihy.cpress.cz/K2223* po klepnutí na odkaz Soubory ke stažení.

KAPITOLA 1

Úvod do jazyka CSS3

V této kapitole:

- Co je CSS3 a jak vzniklo
- Stavy modulů a proces doporučování
- Představení syntaxe
- Proprietární předpony
- Začněme

V první kapitole si popíšeme konvence zápisu kódu použité v této knize a povíme si něco o syntaxi, která je jedinečná pro jazyk CSS3. Nejprve si ale uvedeme několik informací z historie tohoto jazyka. Znalost historie samozřejmě není nutná, ale myslím si, že historické souvislosti jsou důležité k pochopení současného stavu tohoto jazyka.

CSS3 je specifikace, jež se neustále mění. Některé její části můžeme považovat za stabilní a jsou již dobře implementované v moderních webových prohlížečích; jiné části bychom měli považovat za experimentální – ty jsou jen částečně implementované. Další části jsou jen ve fázi návrhu a nebyly dosud implementované vůbec. Někteří výrobci prohlížečů vytvořili své vlastní vlastnosti jazyka CSS, které nejsou součástí specifikace a pravděpodobně nikdy nebudou.

Z toho všeho plyne, že znalost toho, jak funguje standardizační proces, a znalost úrovní implementace nových vlastností jsou nezbytné k tomu, abychom věděli, co ze specifikace CSS3 můžeme používat v našem kódu už dnes a co budeme moct používat v budoucnosti.

Co je CSS3 a jak vzniklo

Pro začátek si ukažme, co CSS3 je, a co naopak není. Přístup konsorcia W3C ke specifikaci CSS3 se značně liší od přístupu ke specifikaci CSS2. Tento přehled by měl pomoct pochopit, jak a kdy používat jazyk CSS3 a proč se tolik liší implementace napříč různými prohlížeči.

Stručná historie CSS3

Poslední hlavní verzí specifikace jazyka CSS byla specifikace CSS2.1. Jednalo se o revizi specifikace CSS2, která vznikla roku 1997. Navzdory neustálému vývoji a množství změn za tu dobu je spousta lidí překvapená, že CSS2 se stala „oficiálním" doporučením konsorcia W3C

v roce 2011 (o procesu vydávání doporučení si povíme za malou chvíli). Mnohem překvapivější je skutečnost, že prohlížeč Internet Explorer 8, vydaný v roce 2009, prohlašoval, že je první prohlížečem s úplnou podporou specifikace CSS2.1.

V posledních několika letech se vývojáři baví o nové verzi – CSS3. Slovo „nové" zde možná není na místě, protože práce na specifikaci CSS3 začaly v roce 1998 – tj. pouhý roku po vydání specifikace CSS2. Implementace specifikace CSS2 v prohlížečích byla ale značně nekonzistentní, a proto se konsorcium W3C rozhodlo přerušit vývoj nové verze a raději se věnovalo specifikaci CSS2.1, která standardizovala proces implementace jazyka CSS ve skutečném světě. V roce 2005 se vrátily všechny moduly specifikace CSS3 do stavu Working Draft a znovu začal proces úprav a hodnocení.

Po dlouhou dobu se rozšiřovala uživatelská základna prohlížeče Internet Explorer, který nevykazoval žádné známky, že by se chystal implementovat specifikaci CSS3. V průběhu posledních deseti let se ale objevila celá řada nových prohlížečů a začal konkurenční boj, který vyústil v závod o co nejrychlejší zavádění nových funkcí. Hlavní výhodou tohoto závodu byla implementace specifikace CSS3. Všichni výrobci prohlížečů chtěli vývojářům a uživatelům přinést nejnovější webové technologie, a jelikož specifikace CSS3 byla z převážné části napsaná, zavádění nových funkcí bylo snadné.

A tak jsme se ocitli až v dnešní době, v níž probíhá aktivní vývoj specifikace CSS3, většina prohlížečů ji implementuje a komunita nadšených vývojářů na ní staví, studuje ji a píše o ní. To je výborná situace, kterou bychom před pár lety jen stěží předpovídali.

CSS3 je modulární

Vytvořit jazyk pro stylování všech značkovací jazyků na světě je nesmírně těžká úloha. Konsorcium W3C vědělo, že může trvat roky, než přinese ovoce. Jeho členové věděli, že bude nutné oddělit srozumitelné funkce od těch tajemnějších, a proto rozdělili specifikaci CSS3 na moduly. Na jednotlivých modulech mohli pracovat různí autoři na různých místech.

Z tohoto důvodu nemáme jeden obrovský dokument specifikace CSS3, ale máme dokumenty CSS3 Basic User Interface Module, Selectors Level 3, Media Queries atd. Některé moduly jsou revizemi ze specifikace CSS2.1, kdežto jiné jsou zcela nové, ale všechny spadají pod specifikaci CSS3.

Jednou věcí, která mě rozčiluje, je, že si lidé stěžují: „Chci používat jazyk CSS3, ale nebude připraven ještě dlouho." To je nesmysl, některé moduly jsou již stabilní a podporují je všechny moderní prohlížeče a jiné dělí od jejich slávy jen několik měsíců. Pokud chcete čekat, až budou moduly 100procentně implementované v prohlížečích, budete čekat věčnost.

Takže specifikace CSS3 je zde, přičemž její část můžete používat už dnes. Musíte si rozmyslet, jak ji budete používat.

CSS3 neexistuje

Jedná se o velmi provokativní nadpis, ale technicky je pravdivý. Protože jazyk CSS se stal modulárním, každý modul je označen číslem úrovně, které určuje, kolika revizemi prošel. Některé vyspělejší moduly, například Selectors, jsou již označeny jako Level 4; spousta modulů popisovaných v této knize je značena Level 3, zatímco nové moduly, například Flexbox, jsou jen ve fázi Level 1 nebo přecházejí do verze Level 2.

To znamená, že CSS je živý standard, přičemž nebudou vznikat žádné další monolitické verze, ale každý modul se bude vyvíjet vlastním tempem. Nové moduly budou přidávány jako nové funkce zcela samostatně. CSS lze považovat tedy za zkratku pro: „funkce jazyka CSS, které se změnily od specifikace 2.1". CSS4 nikdy nevznikne. Případně se časem upustí od číslování úplně a zůstane nám jen jazyk CSS s moduly různých úrovní.

Nenechme se ale odradit. V této knize budeme nadále mluvit o CSS3 v tom smyslu, který jsme si definovali výše. Tento termín usnadňuje vysvětlování – a také nebudu muset měnit název této knihy.

Stavy modulů a proces doporučování

Na řadě míst v této knize můžete narazit na termín **stav modulu**. Stav modulu určuje konsorcium W3C a označuje, v jaké fázi procesu doporučování se daný modul nachází. Zapamatujte si však, že stav není označení pro stupeň implementace modulu v prohlížečích.

Když konsorcium W3C přijme navržený dokument jako součást specifikace CSS3, přidělí mu stav „Working Draft". To znamená, že dokument byl publikován a je připraven na hodnocení komunity – v tomto případě ji tvoří výrobci prohlížečů, pracovní skupiny a další lidé, kteří se o to zajímají. Dokument může být v tomto stavu velmi dlouho a může projít mnoha revizemi. Ne všechny dokumenty tento stav překonají. A také se dokument může z mnoha důvodů do tohoto stavu vrátit.

Když je dokument připraven na opuštění stavu Working Draft, změní se jeho stav na „Last Call", což znamená, že období hodnocení končí a dokument může postoupit do další úrovně.

Další úrovní je „Candidate Recommendation". V tomto stavu je již konsorcium W3C spokojeno, protože dokument dává smysl. Poslední komentáře neodhalily žádné závažné problémy a jsou splněny všechny technické požadavky. V tomto okamžiku můžou výrobci prohlížečů začít implementovat nové vlastnosti, aby získali odezvu ze skutečného světa.

V případě, že dva prohlížeče implementovaly dané vlastnosti stejným způsobem a neobjevily se žádné vážné technické problémy, dokument může přejít do stavu „Proposed Recommendation". To znamená, že návrh dozrál a dočkal se implementace, a tudíž ho může schválit výbor W3C Advisory Committee. Jakmile ten udělí souhlas, z návrhu se stane doporučení – je ve stavu „Recommendation".

Zopakujeme, co jsme nakousli dříve: Proces doporučování a implementační proces nefungují vždy stejně. Modul může být implementovaný napříč všemi prohlížeči a přitom být ve stavu Working Draft – v době psaní této knihy má přesně takový stav modul Transitions (o němž si povíme v kapitole 14). Nebo modul může být naopak ve stavu Candidate Recommendation a mít jen omezenou implementaci – v současnosti tomuto popisu vyhovuje modul CSS Shapes (více se dozvíte v kapitole 19).

Proto jsem uspořádal tuto knihu podle úrovně implementace, a ne stavu doporučování. V dřívějších kapitolách si popíšeme funkce, které mají plnou úroveň implementace ve všech webových prohlížečích (nebo by měly mít v době vydání této knihy). Pozdější kapitoly se věnují vlastnostem, které jsou implementované jen v některých prohlížečích a obvykle mají proprietární předpony. Kapitoly na konci této knihy se zabývají jen potenciálními nebo částečnými implementacemi vlastností.

Představení syntaxe

Úvod máme za sebou, takže si pojďme představit jádro jazyka CSS3. V této knize používám určité syntaktické konvence pro demonstraci nových pravidel a vlastností. Vypadá přibližně takto:

```
E { vlastnost: hodnota; }
```

Ve výše uvedeném ukázkovém kódu značíme selektor písmenem `E`. V jazyce HTML nemá takový selektor samozřejmě žádný ekvivalent. Jedná se jen o příklad, za který si můžeme dosadit cokoliv.

Následuje samotná vlastnost, přičemž v tomto příkladu jsme použili obecný název `vlastnost`. Za názvem vlastnosti se nachází hodnota – nyní je místo ní opět zástupný alias `hodnota`.

Pokud vlastnost přijímá více hodnot, uvedeme si je s jedinečnými aliasy. Novou vlastnost vyžadující tři hodnoty bychom mohli definovat takto:

```
E { vlastnost: první druhá třetí; }
```

Představme si, že by existovala vlastnost `opice` (vždy jsem chtěl vlastnost `opice`), která přijímá jedinou hodnotu. Podle syntaxe používané v této knize bychom si ji představili takto:

```
E { opice: hodnota; }
```

Kdybychom chtěli praktický příklad, mohli bychom si ukázat platnou hodnotu – například číselnou:

```
E { opice: 12; }
```

Proprietární předpony

V případě, že probíhá aktivní hodnocení modulu, může se mnohé změnit – včetně syntaxe vlastnosti, nebo může dokonce celá vlastnost zmizet. Někdy se také stává, že je formulace samého návrhu nejasná a nelze ho jednoznačně interpretovat.

Prohlížeče ale implementují tyto nové funkce, takže můžeme zjistit, jak fungují v praxi. Nesmíme ale zapomínat na to, že dva různé prohlížeče můžou implementovat stejnou vlastnost jinak – zobrazovaný výsledek se v nich bude lišit. Aby tomu výrobci prohlížečů zabránili, začali přidávat krátkou předponu na začátek experimentálních vlastností.

Představme si, že naše vlastnost `opice` se objevila ve specifikaci a výrobci se ji rozhodli implementovat, aby zjistili, jak funguje. V takovém případě bychom použili níže uvedený kód:

```
E {
  -moz-opice: hodnota; /* Firefox */
  -ms-opice: hodnota; /* Internet Explorer */
  -webkit-opice: hodnota; /* Chrome/Safari */
}
```

Opakování podobných deklarací vlastností se může zdát zbytečné, ale je pro naše dobro. Poslední věcí, o kterou bychom stáli, by bylo, kdyby všechny prohlížeče implementovaly vlastnost `opice` jinak, což by vedlo k naprostému chaosu.

Třebaže předpony proprietárních vlastností vznikly z dobrého úmyslu, jejich používání vede k řadě problémů. Vývojáři je používají v produkční verzi svých stránek, ale neodstraňují je, když se změní implementace v prohlížečích. To na druhou stranu vede k tomu, že výrobci prohlížečů nadále podporují experimentální funkce, aby neporušili funkčnost některých webových stránek. Z tohoto důvodu prohlížeče Chrome a Firefox upouštějí od proprietárních vlastností a raději implementují nové funkce jako zakázané a vývojáři si je musejí povolit, než se stanou dostatečně stabilní. Proprietárních vlastností existuje ale stále spousta a tato kniha upozorňuje na to, kdy je nutné je použít.

Začněme

To je vše, co potřebujete, abyste mohli bez obav začít číst tuto knihu – kromě zvídavé povahy. Musím toho říct o jazyce CSS3 opravdu hodně, proto se budu přesouvat mezi tématy rychleji, ale ve všech kapitolách byste měli získat dost znalostí, abyste mohli vytvářet vlastní testy, příklady a webové stránky, ve kterých budete používat flexibilitu a bohaté funkce tohoto jazyka.

Začneme nejjednodušší a přitom převratnou funkcí – dotazy na médium.

KAPITOLA 2

Dotazy na médium

V této kapitole:

- Výhody dotazů na médium
- Syntaxe
- Vlastnosti média
- Více vlastností média
- Webový vývoj mobile-first
- Shrnutí
- Dotazy na médium – podpora v prohlížečích

Když uživatelé přistupovali k webu jen z prohlížeče na stolním počítači nebo notebooku, psát kaskádové styly bylo poměrně jednoduché. Třebaže jsme se museli vypořádat s rozdíly mezi prohlížeči a platformami, alespoň jsme věděli, že všichni prohlížejí naše webové stránky na podobných zařízeních. V posledních letech se ale s novými zařízeními s přístupem k webu doslova roztrhl pytel – od herních konzolí k mobilním zařízením, jako jsou chytré telefony nebo tablety. Prezentování obsahu všem stejným způsobem už nedává smysl, protože uživatelé ho prohlíží na širokých monitorech stolních počítačů, ale i na úzkých příručních obrazovkách.

Jazyk CSS již dlouhou dobu umožňuje poskytovat různé styly pro různé typy médií, a to pomocí atributu `media` elementu `link`:

```
<link href="styly.css" rel="stylesheet" media="screen">
```

Tento přístup má spoustu nedostatků – jedná se o nástroj, který se musíte naučit používat pro obrazovky o úhlopříčce mezi 3,5 palci a 32 palci. Nabídka typů medií je příliš široká a některé nejsou podporované ani zařízeními, pro něž jsou určené – neznám kupříkladu žádnou televizi s podporou webu, která by reagovala na typ `tv`. Dle očekávání začalo konsorcium W3C odrazovat od používání typů médií.

Specifikace CSS3 přišla s řešením – s **dotazy na médium**, které jsou definované v modulu Media Queries (*http://www.w3.org/TR/css3-mediaqueries/*). Tyto dotazy rozšiřují typy médií o syntaxi, s níž lze přesněji specifikovat zařízení uživatele, a tak uživatelům zprostředkovat požitek uzpůsobený na míru. Tento popis může znít poněkud suše, ale tato funkce je ve skutečnosti jednou z nejrevolučnějších novinek specifikace CSS3. S dotazy na médium získáte svobodu pro tvorbu webových stránek, jež jsou skutečně nezávislé na zařízení. Tím uživatelům dáte nejlepší prožitek bez ohledu na to, odkud vaše stránky navštěvují.

Modul Media Queries je ve stavu Recommendation, takže je považován za standard. Tento modul je již dobře implementován ve většině hlavních webových prohlížečů, a to včetně prohlížeče Internet Explorer od verze 9.

Výhody dotazů na médium

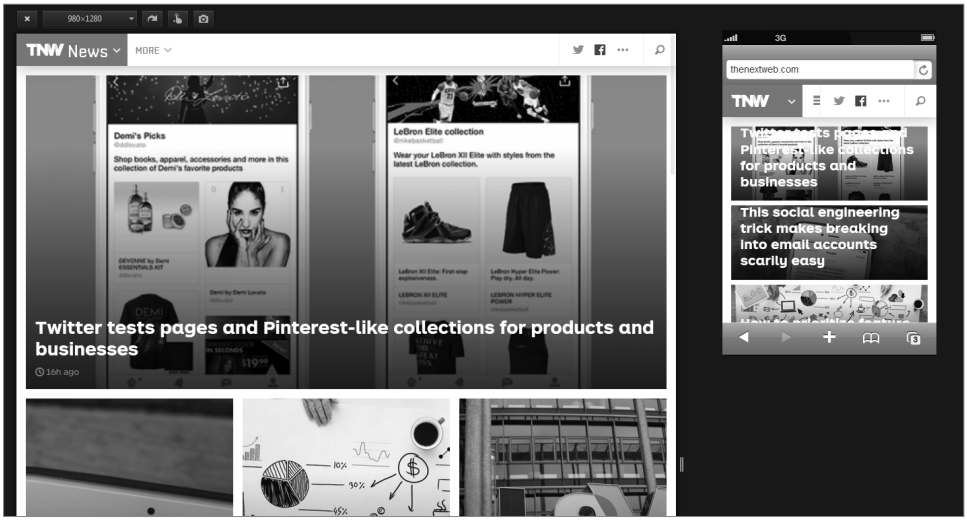
Jako rychlou ukázku síly a flexibility dotazů na médium si ukážeme příklad toho, jak lze optimalizovat stránky pro mobilní prohlížeče, aniž by bylo nutné psát spoustu dodatečného kódu.

Lidé navštěvující naše stránky můžou mít problémy při přístupu z mobilních zařízení – text může být příliš malý a přibližování by vyžadovalo spoustu posouvání při hledání navigačních prvků. K těmto navigačním prvkům můžou patřit i rozevírací seznamy, které se otevírají po přesunutí ukazatele myši nad ně, což je akce, která na mobilních zařízeních nenastává. Stahování velkých obrázků může trvat příliš dlouho na pomalém připojení a může vyčerpat měsíční datový limit.

Některé webové stránky poskytují mobilní verze, ale jejich vývoj je náročný. Je nutné nastavit subdoménu se šablonami stylů a šablonami v jazyce HTML, které se liší od desktopové verze stránek, a také vytvořit skript, který bude detekovat, zda se jedná o mobilní prohlížeč, aby přesměroval na správnou verzi. Toto řešení trpí několika problémy – musíme aktualizovat skript pro všechny verze mobilních prohlížečů a údržba takových stránek často vyžaduje, abychom dělali změny v mobilní verzi a desktopové verzi současně.

Dotazy na médium řeší spoustu těchto problémů. Například detekují zařízení podle jejich vlastností, takže už není nutné programovat skripty pro detekci prohlížečů. Umožňují cílit šablony stylů přímo na schopnosti zařízení. Pokud má zařízení malou obrazovku, kaskádové styly se jí přizpůsobí – odstraní nadbytečné elementy, nabídnou menší obrázky a přizpůsobí text.

Prohlédněte si například technologické webové stránky The Next Web (*http://thenextweb.com/*) na obrázku 2.1.



Obrázek 2.1 Webové stránky Next Web zobrazené na obrazovce stolního počítače a v mobilní verzi (vložené okno)

Když zobrazíme tyto stránky v desktopovém prohlížeči, bude viditelná dlouhá horní navigace a hlavní obsah rozvržený do mřížky. Při zobrazení stránek na užší obrazovce (například na telefonu iPhone) bude – díky síle dotazů na médium – navigace kompaktnější, související obsah bude skrytý a hlavní obsah bude součástí jediného sloupce.

Uživatelé samozřejmě nenavštěvují webové stránky pouze ze stolních počítačů a chytrých telefonů a měli bychom se snažit, aby webové stránky byly optimalizované pro všechna zařízení. Více informací je k dispozici v části „Responzivní webdesign".

Kdybyste se chtěli dozvědět, co dělají s dotazy na médium ostatní vývojáři, prohlédněte si úžasnou galerii na internetové adrese *http://mediaqueri.es/*. Ta obsahuje vynikající příklady, co je možné.

Syntaxe

Dotaz na médium definuje parametr (nebo skupinu parametrů), který povolují přidružená pravidla stylů, pokud vlastnosti zařízení použitého k prohlížení aktuální stránky odpovídají tomuto parametru. Dotazy na médium je možné používat třemi způsoby, přičemž ty přesně kopírují různé způsoby, jimiž lze přidělit kaskádové styly k dokumentu. Můžeme se kupříkladu odkázat na externí šablonu stylů pomocí elementu `link`:

```
<link href="soubor" rel="stylesheet" media="logika médium and (výraz)">
```

Responzivní webdesign

V roce 2010 napsal Ethan Marcotte článek „Responsive Web Design" (*www.alistapart.com/articles/responsive-web-design/*), v němž sjednotil nápady na tvorbu webových stránek, které se umí přizpůsobovat použitím zařízení, a to díky síle dotazů na médium. V něm také prohlásil:

„Přišel čas, kdy navrhujeme obsah tak, aby jej bylo možné prohlížet na široké škále zařízení. Responzivní webdesign představuje obrovský pokrok – konečně nám umožňuje navrhovat věci přirozeně podle toku obsahu."

Od té doby se stal responzivní webdesign standardem. Převážná část vývojářů uvažuje tímto způsobem a každoročně vytvářejí nebo předělávají spoustu webových stránek pomocí metod responzivního webdesignu. Tento druh návrhu přináší také spoustu výzev – navrhování fluidních responzivních stránek muselo projít řadou změn, protože většina návrhářských nástrojů s tím nepočítala. Můžeme ale s klidem říct, že jsme na nejlepší cestě k tvorbě webu, který bude přístupný všem, kdekoliv a na jakémkoliv zařízení.

Druhý způsob spočívá v odkazování na externí šablonu stylů prostřednictvím direktivy `@import`:

```
@import url('soubor') logika médium and (výraz);
```

Třetí způsob je použít dotaz na médium ve vloženém elementu `style` nebo v samotné šabloně stylů pomocí rozšířeného pravidla `@media`:

```
@media logika médium and (výraz) { pravidla }
```

Tuto metodu budeme používat po zbytek této kapitoly, protože je nejčistější a hodí se pro demonstrační účely. Jakou metodu máte zvolit, závisí na vašich osobních preferencích a na požadavcích současné struktury šablon stylů.

Když jsme si představili deklarační metody, prozkoumejme ještě syntaxi. Atribut `media` je pravděpodobně všem známý – definuje typ média, na které se aplikují dané kaskádové styly. Může být součástí elementu `link`:

```
<link href="styly.css" rel="stylesheet" media="screen">
```

Nejrozšířenější typy média jsou `screen` a `print`, přičemž je možné oddělovat více hodnot čárkami (ačkoliv to už není ani nutné, jelikož ostatní typy médií pomalu ale jistě mizí). Pokud neuvedeme typ média, prohlížeč dosadí výchozí typ `all`. Kdybychom tedy psali pravidla stylů pro všechna zařízení, nemusíme uvádět typ média v dotazu na médium. Níže uvedené příklady fungují stejně:

```
@media all and (výraz) { pravidla }
@media (výraz) { pravidla }
```

Poznámka: Aby byly ukázkové kódy ve zbytku knihy výstižnější, vynechám typ média tam, kde není nutný.

Prvním novým parametrem pravidla `@media` je logika. Tento volitelný parametr může mít jako hodnotu klíčové slovo `only` nebo `not`:

```
@media only médium and (výraz) { pravidla }
@media not médium and (výraz) { pravidla }
```

Hodnota `only` je užitečná převážně tehdy, když chceme skrýt pravidlo stylu před staršími prohlížeči, které nepodporují tuto syntaxi. Výraz `not` neguje daný dotaz na médium – aplikuje styly v případě, že nejsou splněny nastavené parametry.

Když použijeme v dotazu parametr `logika` nebo `médium`, musíme použít také operátor `and` jako ve výše uvedených příkladech, abychom k nim připojili `výraz`. Výraz nám umožňuje nastavit různé další parametry, ne jen typ média. Tyto parametry nazýváme vlastnosti média. Právě ty dodávají dotazům na médium jejich sílu. Proto je nyní podrobněji prozkoumáme.

Vlastnosti média

Vlastnosti média jsou informace o zařízení, které zobrazuje dané webové stránky – rozměry, rozlišení atd. Tyto informace slouží k vyhodnocení `výrazu`, přičemž výsledek ovlivní, která pravidla stylů se aplikují. Výraz může kupříkladu říkat: „Aplikuj tyto styly jen na zařízení s obrazovkou širší než 480 pixelů." Nebo: „Aplikuj je jen na zařízení, která jsou otočená vodorovně."

Většina výrazů s vlastnostmi média vyžaduje, abychom uvedli hodnotu:

```
@media (vlastnost: hodnota) { pravidla }
```

Tato hodnota je nezbytná pro stavbu výrazů, o nichž jsme se bavili. Ve výjimečných případech můžeme vynechat hodnotu a testovat pouze existenci samotné vlastnosti média:

```
@media (vlastnost) { pravidla }
```

Jak výrazy fungují, bude jasnější, až si ukážeme různé vlastnosti média a vysvětlíme si, kdy jsou hodnoty povinné nebo volitelné.

Když už jsme si popsali syntaxi, představme si několik nejrozšířenějších vlastností média. V prvé řadě si ukážeme vlastnosti, které jsou společné všem barevným obrazovkám určeným pro přístup k webu – tj. těm, které používáme každodenně. Existují rovněž další vlastnosti média, ale s těmi bychom se setkali spíše u alternativních zařízení, jako jsou například televize nebo terminály s pevnou mřížkou (za předpokladu, že je tato zařízení vůbec podporují).

Šířka a výška

Vlastnost média `width` reprezentuje šířku zobrazovacího průhledu média definovaného typu, což většinou odpovídá šířce okna webového prohlížeče (včetně posuvníku) u desktopových operačních systémů. Základní syntaxe vyžaduje délkovou hodnotu:

```
@media (width: 600px) { pravidla }
```

V tomto případě uplatňujeme tato pravidla stylů jen v prohlížečích, které jsou široké přesně 600 pixelů – to je zajisté příliš konkrétní. Vlastnost `width` může mít také některou z předpon `max-` a `min-`, s nimiž je možné ověřovat maximální a minimální šířku:

```
@media (max-width: 480px) { pravidla }
@media (min-width: 640px) { pravidla }
```

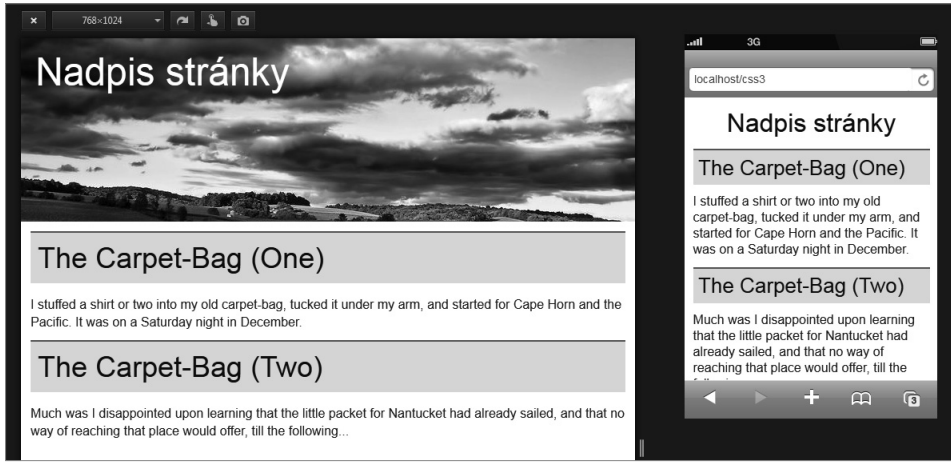
Prvním dotazem na médium aplikujeme příslušná pravidla stylů jen v prohlížečích, které nejsou širší než 480 pixelů, zatímco druhým dotazem na médium je uplatníme pouze v prohlížečích, které jsou alespoň 640 pixelů široké.

Ukažme si praktický příklad. Využijeme větší velikosti okna webového prohlížeče tak, že poskytneme uživatelům okrasný nadpis (pro zachování jednoduchosti vypustíme některá pravidla stylů):

```
@media (max-width: 400px) {
  h1 { background: url('krajina.jpg'); }
}
```

Pomocí výše uvedeného dotazu na médium vybíráme jen prohlížeče s průhledem, který je široký přinejmenším 400 pixelů, a nastavujeme obrázek na pozadí elementům `h1`.

Pokud bude mít okno našeho prohlížeče šířku alespoň 400 pixelů, uvidíme tento obrázek. Kdybychom ho zúžili pod tuto hranici, zobrazil by se pouze text nadpisu. Příklad lze vidět na obrázku 2.2.



Obrázek 2.2 Různá pravidla stylů aplikovaná pomocí vlastnosti média width zobrazená v desktopovém počítači a na mobilu

Vlastnost média `height` funguje podobně – s tím rozdílem, že vybírá prohlížeče podle jejich výšky, a ne šířky. Syntaxe je stejná jako u vlastnosti `width`, přičemž jsou k dispozici i předpony `max-` a `min-`:

```
@media (height: hodnota) { pravidla }
@media (max-height: hodnota) { pravidla }
@media (min-height: hodnota) { pravidla }
```

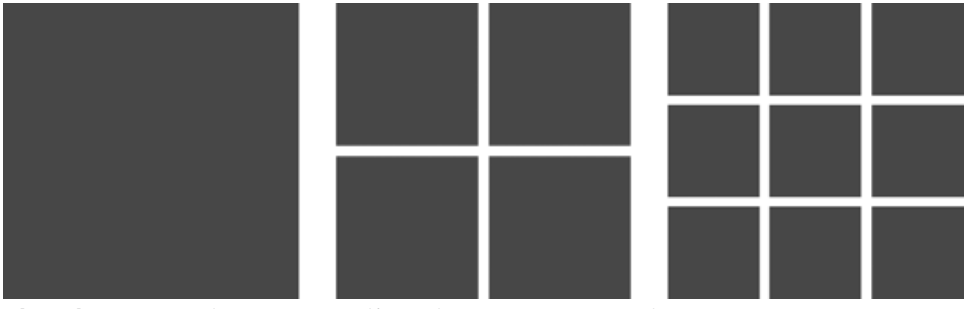
Protože na webu značně převažuje svislé posouvání obsahu, vývojáři nepoužívají vlastnost `height` zdaleka tak často jako vlastnost `width`.

Poměr pixelů

Jednotka pixel v jazyce CSS odpovídá pixelu na počítačové obrazovce. Jestliže průhled má šířku 1 024 pixelů a elementu nastavíme šířku 1 024 pixelů, očekáváme, že ve vodorovném směru vyplní celý průhled. Spousta nových zařízení, zejména chytré telefony a tablety, má obrazovku se super vysokým rozlišením. Element o šířce 1 024 se na nich může jevit jako malý a obtížně čitelný.

Tato novější zařízení často rozlišují CSS pixely od fyzických pixelů samotného zařízení. Umožnuje přibližovat a oddalovat obsah a také dosáhnout vysoké grafické věrnosti na malých obrazovkách. Poměr mezi fyzickými pixely a CSS pixely označujeme jako **poměr pixelů zařízení** (DPR; z anglického **device pixel ratio**). Telefon iPhone 5S má kupříkladu DPR 2, což znamená, že jeden CSS pixel odpovídá 4 fyzickým pixelům – dvěma vodorovným a dvěma svislým.

Znázorněné to můžete vidět na obrázku 2.3. Vlevo se nachází jeden CSS pixel na „normální“ obrazovce s poměrem pixelů 1:1. Uprostřed vidíte stejný CSS pixel na obrazovce s DPR 2, jakou má například telefon iPhone; jsou zde čtyři fyzické pixely na stejném místě. Vpravo se nachází příklad, jak by vypadal pixel na obrazovce s DPR 3, kterou má kupříkladu telefon Nexus 5. Zde už je 9 fyzických pixelů v jediném CSS pixelu.



Obrázek 2.3 CSS pixel s poměrem pixelů 1:1 (vlevo), s DPR 2 (uprostřed) a 3 (vpravo)

V praxi to znamená, že ačkoliv telefon iPhone 5S má fyzické rozlišení 640 × 1136, jeho CSS rozlišení je 320 × 568. To je přesně polovina, protože každý CSS pixel odpovídá dvěma fyzickým pixelům, a to jak vodorovně, tak svisle (ale jen za předpokladu, že je toto zařízení v **mobilním režimu** – více informací obsahuje část „Šířka a výška zařízení“).

Třebaže díky velkému DPR je škálovatelný obsah (například text nebo vektorová grafika) ostřejší a čistější, bitmapové obrázky můžou utrpět výraznou ztrátu kvality, když si je uživatelé prohlížejí na obrazovkách s vysokým rozlišením. Abychom mohli vyřešit tento problém, vznikla nová vlastnost média `resolution`, pomocí níž můžeme cílit na zařízení podle jejich DPR:

```
@media médium and (resolution: hodnota) { pravidla }
```

Hodnotou vlastnosti `resolution` je číslo s jednotkou rozlišení – **bodů na palec** (DPI; dots per inch), **bodů na centimetr** (DPCM; dots per centimeter) nebo pro nás nejdůležitější jednotkou **bodů na pixel** (DPPX; dots per pixel). Jednotka DPPX odpovídá DPR zařízení. Kdybychom tedy chtěli aplikovat pravidla stylů jen na zařízení s DPR 1,5, napsali bychom:

```
@media (resolution: 1.5dppx) { pravidla }
```

Stejně jako u ostatních vlastností média můžeme rovněž detekovat maximální a minimální poměr pixelů:

```
@media (max-resolution: číslo) { pravidla }
@media (min-resolution: číslo) { pravidla }
```

Tato flexibilita umožňuje nabízet obrázky s vyšším rozlišením pro zařízení s vyšší hustotou pixelů, jak lze vidět na níže uvedeném kódu:

```
E { background-image: url('obrazek-niroz.png'); }
@media (min-resolution: 1.5dppx) {
  background-image: url('obrazek-vyroz.png');
  background-size: 100% 100%;
}
```

Prvním pravidlem stylu nastavujeme běžný obrázek (*obrazek-niroz.png*) pro zařízení s „běžným" poměrem pixelů (s nižším rozlišením), zatímco pro zařízení s DPR alespoň 1,5 použijeme obrázek s vyšším rozlišením (*obrazek-hires.png*). Používáme zde také neznámou vlastnost `background-size`; tu bychom měli používat u obrázků s vyšším rozlišením, aby je prohlížeč nezobrazoval větší než element, jemuž obrázek nastavujeme jako pozadí (tuto vlastnost si podrobně popíšeme v kapitole 8).

Prohlížeče Chrome, Firefox a Internet Explorer 10+ podporují vlastnost média `resolution`, ale prohlížeč IE zatím neimplementuje jednotku DPPX. Měli bychom v něm používat jednotku DPI a násobit DPR číslem 96 (hodnota DPI u běžné obrazovky). Zde je příklad:

```
@media (resolution: 1.5dppx), (resolution: 144dpi) { pravidla }
```

Prohlížeč Safari nepodporuje vlastnost `resolution`, ale podporuje proprietární vlastnost média `-webkit-device-pixel-ratio` (i varianty `max-` a `min-`), jež přijímá jako hodnotu jediné číslo bez jednotky, které odpovídá požadovanému DPR. Řešení pro všechny moderní webové prohlížeče by tudíž vypadalo takto:

```
@media (resolution: 1.5dppx), (resolution: 144dpi),
  (-webkit-device-pixel-ratio: 2) { pravidla }
```

Jádro WebKit zavedlo vlastnost `resolution` na konci roku 2012, proto je překvapující, že prohlížeč Safari ji neimplementoval v době, kdy jsem psal tuto knihu – téměř o dva roky později. Doufám, že to společnost Apple brzy napraví.

Šířka a výška zařízení

Vlastnosti `width` a `height` se vztahují k rozměrům průhledu prohlížeče, ale průhled nemusí mít vždy stejné rozměry jako obrazovka. Kdybyste chtěli cílit na rozměry obrazovky, mohli byste použít vlastnosti `device-width` a `device-height`; případně i s předponami `min-` a `max-`. Ty nebudete však používat příliš často, ale abych vysvětlil proč, musím na chvíli odbočit.

V minulé části této kapitoly jsme si vysvětlili rozdíl mezi CSS pixely a fyzickými pixely. Vlastnost `width` měříme v CSS pixelech, kdežto vlastnost `device-width` ve fyzických pixelech. Aby byl obsah čitelný a přirozeně velký na malých obrazovkách, musí si oba rozměry odpovídat. Toho dosáhneme doplněním značky `meta` s názvem `viewport` do záhlaví stránky (do elementu `head`):

```
<meta name="viewport" content="width=device-width">
```

Když mobilní prohlížeč narazí na tuto značku v záhlaví stránky, přepne do **mobilního režimu**, v němž průhled získá ideální rozměry pro aktuální zařízení. Obsah stránky bude mít tedy lepší velikost odpovídající aktuálnímu zařízení.

Poznámka: Detailnější popis průhledů a pixelů mobilních zařízení najdete v článku „A Pixel Is Not a Pixel Is Not a Pixel" na internetové adrese *http://www.quirksmode.org/blog/archives/2010/04/a_pixel_is_not.html*.

Průhled prohlížeče na mobilních zařízeních bývá stejně velký jako samotná obrazovka, takže tyto dva rozměry bývají totožné. V desktopových prohlížečích se obvykle nastavuje velikost obsahu relativně k průhledu, a ne k obrazovce. Z těchto důvodů se stává vlastnost média `device-width` méně užitečnou než vlastnost `width` a v praxi ji příliš nepoužijete.

Objevila se rovněž možnost standardizovat značku `meta` s názvem `viewport` také v jazyce CSS, a to v podobě pravidla `@viewport`.

Orientace

Pokud vás tolik nezajímají skutečné rozměry zobrazovacího zařízení, ale chcete optimalizovat své stránky pro vodorovné (typický desktopový/notebookový webový prohlížeč) nebo svislé prohlížení, přijde vám k užitku vlastnost média `orientation`. Následuje syntaxe:

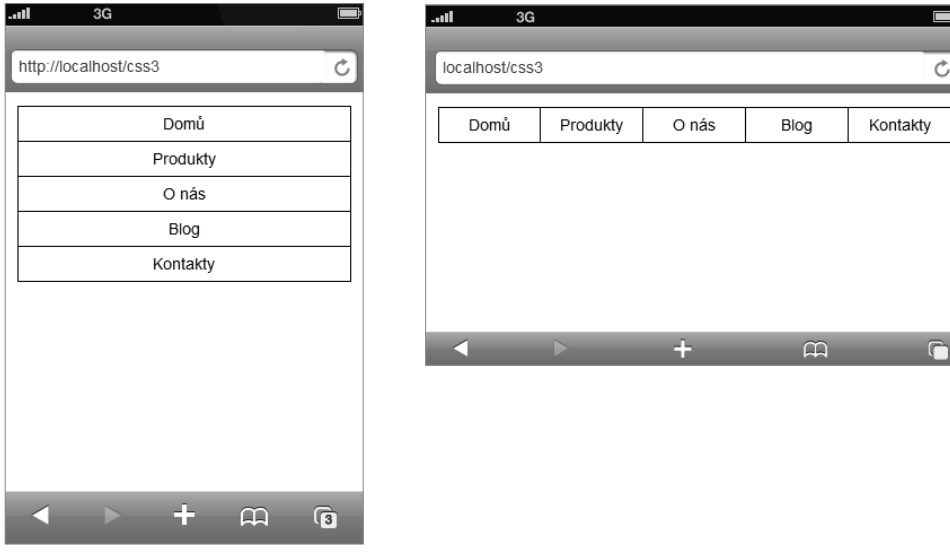
```
@media (orientation: hodnota) { pravidla }
```

Hodnotou může být některé ze dvou klíčových slov – `landscape` nebo `portrait`. Hodnota `landscape` značí, že šířka webového prohlížeče je větší než výška, zatímco hodnota `portrait` označuje přesný opak. Ačkoliv lze vlastnost `orientation` používat i v desktopových prohlížečích, nejužitečnější je u příručních zařízení, která uživatel může otáčet; například u chytrých telefonů a tabletů.

S vlastností `orientation` můžeme kupříkladu zobrazovat vodorovnou nebo svislou navigační nabídku v závislosti na otočení. Zdrojový kód by mohl vypadat následovně:

```
ul { overflow: hidden; }
li { float: left; }
@media (orientation: portrait) {
  li { float: none; }
}
```

Standardně budou elementy `li` zarovnané vlevo, takže se budou na stránce skládat vedle sebe. Když budeme tuto stránku prohlížet při otočení `portrait`, ať změnou velikosti okna webového prohlížeče, nebo otočením zařízení do svislé polohy, odstraníme zarovnání vlastností `float`, takže se elementy `li` budou skládat svisle pod sebe. Výsledek lze vidět na obrázku 2.4.



Obrázek 2.4 Vlastnost média orientation v mobilním prohlížeči – portrait (vlevo) a landscape (vpravo)

Jelikož vlastnost `orientation` přijímá pouze jednu ze dvou hodnot, když aplikujeme pravidla stylů pomocí některé hodnoty, ostatní se uplatní v opačném případě. V tomto příkladu jsme použili jen hodnotu `portrait`, takže ostatní pravidla stylů budou platit při otočení `landscape`.

Poměr stran

Můžeme rovněž vytvářet dotazy na médium pro určité poměry šířky a výšky. Pomocí vlastnosti `aspect-ratio` můžeme testovat poměr stran průhledu nebo pomocí vlastnosti `device-aspect-ratio` ověřovat poměr stran zařízení. Takto vypadá syntaxe pro tyto dvě vlastnosti:

```
@media (aspect-ratio: vodorovně/svisle) { pravidla }
@media (device-aspect-ratio: vodorovně/svisle) { pravidla }
```

Hodnoty *vodorovně* a *svisle* jsou kladná celá čísla, která reprezentují poměr šířky a výšky obrazovky, takže čtvercová obrazovka má poměr 1/1 a širokoúhlá obrazovka má poměr 16/9.

Poznámka: Některá zařízení (zejména iPhone) vždy oznamují poměr stran při otočení zařízení ve svislé poloze, dokonce i tehdy, když je otočíme vodorovně.

Při výběru podle poměru stran je nutné dbát zvýšené opatrnosti. Například někteří výrobci zařízení definují širokoúhlé obrazovky jako 16/9, jiní jako 16/10 a další jako 15/10. Zařízení také nemusí mít deklarovaný poměr stran. Například iPhone 5S prohlašuje, že má poměr stran 16/9, ale prohlížeči oznamuje mírně větší poměr 40/71 (ve svislém otočení). Proto je lepší používat i předpony `max-` a `min-` u vlastností `aspect-ratio` a `device-aspect-ratio`. Například takto bychom mohli aplikovat pravidla stylů, pokud má zařízení poměr stran alespoň 16/9:

```
@media (min-device-aspect-ratio: 16/9) {...}
```

Více vlastností média

Můžeme řetězit více dotazů na médium pro stejný typ média přidáním výrazů a operátoru `and`:

```
@media logika médium and (výraz) and (výraz) { pravidla }
```

Tímto zápisem ověřujeme oba výrazy, než aplikujeme vybraná pravidla. Například úzkou obrazovku s maximálním poměrem stran 15/10 bychom otestovali pomocí tohoto dotazu:

```
@media (max-device-aspect-ratio: 15/10) and (max-width: 800px) {...}
```

Můžeme rovněž použít operátor `or` tak, že dodatečné dotazy zapíšeme do seznamu s čárkami:

```
@media logika médium and (výraz), logika médium and (výraz) { pravidla }
```

Takto aplikujeme příslušná pravidla stylů, když je jakákoliv podmínka pravdivá. V níže uvedeném kódu je aplikujeme na obrazovky otočené vodorovně a při tisku na výšku:

```
@media screen and (orientation: landscape), print and (orientation:
portrait) {...}
```

Samozřejmě je možné vytvořit jakoukoliv kombinaci těchto syntaktických prvků.

Webový vývoj mobile-first

Praxí osvědčenou metodou při stavbě moderních webových stránek je **vývoj mobile-first**, při němž začínáme s vývojem pro malé obrazovky a až potom přidáváme větší prostředky a složitost pro uživatele přistupující z větších zařízení.

Důvodem, proč si vývojáři oblíbili tento přístup, je, že prohlížeče načítají prostředky, které uvedeme v šablonách stylů (například obrázky).

Problémy nastaly, když první uživatelé dotazů na médium (vývojáři) začali nastavovat elementům velké obrázky na pozadí a pak psali pravidla stylů, s nimiž je skryli pro mobilní zařízení:

```
E { background-image: url('velky-obrazek.jpg'); }
@media (max-width: 600px) {
  E { display: none; }
}
```

Ačkoliv jsou tyto obrázky na pozadí skryté, prohlížeč je stahuje a ukládá je do mezipaměti. Tato metoda prodlužuje dobu načítání stránky a spotřebovává datový limit – ani jedno z toho není dobré pro uživatele mobilních zařízení, kteří nemají k dispozici rychlé připojení k Internetu.

Když vytváříme stránky přístupem mobile-first, začínáme od základní šablony stylů, kterou uplatníme ve všech webových prohlížečích (včetně mobilních), a potom postupně přidáváme prostředky a funkce pro uživatele větších obrazovek, přičemž je načítáme dotazem na médium s vlastností `min-width`:

```
@media (min-width: 600px) {
  E { background-image: url('velky-obrazek.jpg'); }
}
```

Tato změna způsobí, že obrázek na pozadí se nikdy nenačte na zařízeních s menšími obrazovkami. Tento přístup lze rozšířit a načítat celé šablony stylů:

```
<link href="zakladni.css" rel="stylesheet">
<link href="desktop.css" rel="stylesheet" media="(min-width: 600px)">
```

Jestliže rozdělíme šablony stylů tímto způsobem, některé webové prohlížeče optimalizují jejich načítání. Protože soubor *`desktop.css`* není nutné načítat pro obrazovky s šířkou do 600 pixelů, například prohlížeč Chrome odloží jeho načítání, dokud nestáhne prostředky s vyšší prioritou – poměrně užitečná optimalizace.

Přístup mobile-first funguje ve většině moderních webových prohlížečů. Opravdu staré prohlížeče si musí vystačit se základní šablonou stylů. To je ale pravděpodobně lepší, pro-

tože si stejně neumí poradit s pokročilejšími vlastnostmi, o nichž se budeme učit ve zbytku této knihy.

Shrnutí

Třebaže mají jednoduchou syntaxi, dotazy na médium můžou být neuvěřitelně užitečné. Díky explozi mobilního webu v uplynulých několika let vývojáři a návrháři zjistili, že umí přizpůsobovat obsah, aniž by museli používat staré techniky detekce webových prohlížečů a tvorby samostatných mobilních verzí.

Rozmach responzivního webdesignu v posledních několika letech byl umožněn vznikem dotazů na médium. Během krátké doby se z nich staly nejužitečnější nástroje každého webového vývojáře. S rozvahou a chytrým zapojením dotazů na médium můžeme vytvářet webové stránky, které se skvěle přizpůsobují uživatelům, ať už přistupují k webu jakkoliv.

Dotazy na médium – podpora v prohlížečích

	Chrome	**Firefox**	**Safari**	**IE**
Dotazy na médium	Ano	Ano	Ano	Ano

KAPITOLA 3

Selektory

V této kapitole:

- Selektory atributů
- Nové selektory atributů ve specifikaci CSS3
- Kombinátor obecného sourozence
- Shrnutí
- Selektory – podpora v prohlížečích

Selektory jsou srdcem jazyka CSS. Specifikace CSS1 jich nabízela jen 5 nebo 6, specifikace CSS2 přinesla 12 nových. Specifikace CSS3 jde ještě dál a zdvojnásobuje počet dostupných selektorů.

Selektory lze rozdělit do dvou kategorií. První z nich fungují přesně pro elementy definované ve stromu dokumentu (například pro elementy `p` nebo atributy `href`). Do této kategorie spadají selektory tříd, typů a atributů. Kvůli zachování jednoduchosti je budeme označovat jako **selektory modelu DOM**. Do druhé kategorie patří **pseudoselektory**, které fungují pro elementy a údaje nacházející se vně stromu dokumentu (například první písmeno odstavce nebo poslední dceřiný element rodičovského elementu). O pseudoselektorech si povíme více v kapitole 4. Nyní se budeme věnovat selektorům modelu DOM.

Specifikace CSS3 přináší tři nové selektory atributů a jeden nový **kombinátor**, což je selektor, jenž spojuje další selektory – například selektor dceřiného elementu (>) ze specifikace CSS2. Ty jsou definované v modulu Selectors Level 3 (*http://www.w3.org/TR/css3-selectors/*), který se stal doporučením konsorcia W3C a dočkal se stabilní implementace napříč prohlížeči.

Pokud nemusíte podporovat prohlížeč Internet Explorer 6, můžete začít používat selektory specifikace CSS3 hned – stejně jako celá řada dnešních webových stránek.

Selektory atributů

Selektory atributů byly zavedeny ve specifikaci CSS2. Jak napovídá jejich název, umožňují specifikovat pravidla stylů pro elementy na základě jejich atributů – například `href` nebo `title` – a hodnot těchto atributů. Specifikace CSS2 definuje čtyři typy těchto selektorů:

```
E[atribut] {...} /* Jednoduchý selektor atributu */
E[atribut='hodnota'] {...} /* Přesný selektor hodnoty atributu */
E[atribut~='hodnota'] {...} /* Částečný selektor hodnoty atributu */
E[atribut|='hodnota'] {...} /* Jazykový selektor atributu */
```

Než přejdeme k novým selektorům specifikace CSS3, zrekapitulujeme si, jak jsou jednotlivé selektory užitečné. Použijeme k tomu níže uvedený kód velmi krátkého seznamu kontaktů:

```
<ul>
  <li><a href="" lang="en-GB" rel="friend met">Peter</a></li>
  <li><a href="" lang="es-ES" rel="friend">Pedro</a></li>
  <li><a href="" lang="es-MX" rel="contact">Pancho</a></li>
</ul>
```

Jednoduchý selektor atributu aplikuje pravidla stylů na elementy, které mají uvedený atribut, a to bez ohledu na jeho hodnotu. Kdybychom tedy měli následující kód:

```
a[rel] { color: red; }
```

vybrali bychom všechny elementy a s atributem `rel`, a to bez ohledu na jeho hodnotu. V tomto případě bychom tudíž uplatnili předchozí pravidlo stylu na všechny naše elementy. Kdybychom chtěli být specifičtější, mohli bychom použít **přesný selektor hodnoty atributu**:

```
a[rel='friend'] { color: red; }
```

Toto pravidlo stylu aplikujeme jen na náš druhý element `a`, jelikož vybíráme pouze elementy s přesnou hodnotou `friend`. V případě, že bychom chtěli vybrat oba elementy s touto hodnotou, museli bychom použít **částečný selektor hodnoty atributu**:

```
a[rel~='friend'] { color: red; }
```

Výše uvedeným selektorem hledáme hodnotu `friend` jako součást seznamu slov odděleného mezerami v atributu `rel`, a proto uplatníme toto pravidlo stylu u prvního a druhého elementu.

Posledním selektorem – **jazykovým selektorem atributu** – hledáme elementy, které mají atribut odpovídající prvnímu argumentu tohoto selektoru a hodnotu, jež odpovídá druhému argumentu a bezprostředně za ní následuje spojovník. Pokud zní tento popis poněkud divně, je tomu tak proto, že jediným účelem tohoto selektoru je hledat dílčí jazykové kódy. Ve výše uvedeném kódu jazyka HTML máme dvě španělská jména, přičemž obě mají atribut `lang` s hodnotou začínající předponou `es-`. Jedno spadá do Španělska (`es-ES`) a druhé do Mexika (`es-MX`). Kdybychom chtěli vybrat oba tyto elementy, napsali bychom:

```
a[lang|='es'] { color: red; }
```

Takovým způsobem vybereme všechny elementy s atributem `lang`, jejichž hodnoty začínají předponou `es-`, a to bez ohledu na jejich kód lokality – v tomto případě vybereme druhý a třetí element `a`. Tento selektor je možné použít pro jakýkoliv atribut s hodnotou, která obsahuje slova oddělená spojovníky, ale většinou ho vývojáři používají pro jazykové kódy.

Poznámka: Zde použité názvy atributů nepocházejí ze specifikace, ale z knihy Erica Meyera s názvem CSS Pocker Reference, kterou vydalo nakladatelství O'Reilly Media v roce 2011.

Nové selektory atributů ve specifikaci CSS3

Zjistili jsme, že se selektory atributů můžeme hledat přesné nebo částečné hodnoty. Ale co kdybychom chtěli více flexibility? Nové selektory specifikace CSS3 umožňují vyhledávat podřetězce v hodnotě atributu. Díky této nové vlastnosti lze pravidla stylů snadněji aplikovat na dokumenty XML, jejichž hodnoty jsou obvykle mnohem rozmanitější než hodnoty dokumentu HTML, ale i tak můžou být užitečné pro vývojáře pracující v jazyce HTML.

Počáteční selektor hodnoty atributu

Prvním novým selektorem, kterému budeme pro jednoduchost říkat **počáteční selektor**, budeme hledat elementy, jejichž hodnota atributu začíná zadaným řetězcem. Používá znak ^ před rovnítkem. Úplný zápis vypadá takto:

```
E[atr^='hodnota'] {...}
```

V tomto kódu hledáme uvedenou hodnotu na začátku daného atributu. Ukažme si praktický příklad se třemi položkami seznamu, z nichž každá obsahuje hypertextový obsah s různými hodnotami atributu `title`:

```
<li><a href="http://priklad.cz/"
  title="obrazová knihovna">Příklad</a></li>
<li><a href="http://priklad.cz/"
  title="volně dostupná obrazová knihovna">Příklad</a></li>
<li><a href="http://priklad.cz/"
  title="volně dostupná zvuková knihovna">Příklad</a></li>
```

Na tento kód aplikujeme následující selektor:

```
a[title^='obrazová'] {...}
```

V tomto případě se aplikuje pravidlo stylu na element `a` v první položce seznamu, protože jeho atribut `title` začíná slovem `obrazová`. Neuplatníme ho ale u elementu `a` v druhé položce, třebaže její atribut `title` rovněž obsahuje toto slovo, jelikož jím nezačíná. Nepoužijeme ho ani u třetí položky seznamu, protože ta vůbec neobsahuje toto slovo.

Poznámka: V dokumentech HTML nezáleží na velikosti písmen hodnoty atributového selektoru, ale v dokumentech XML musíme dodržovat přesnou velikost písmen u těchto hodnot.

Počáteční selektor se hodí zejména na vizuální odlišení hypertextových odkazů. Následuje ukázka typického odkazu směřujícího na externí webové stránky:

```
<p>Toto je <a href="http://priklad.cz/">hypertextový odkaz</a>.</p>
```

Když vidíte odkaz ve svém prohlížeči, nemůžete okamžitě říct, zda se jedná o odkaz na stránku na stejném serveru, nebo o externí adresu URI. Tomuto novému selektoru můžete ale předat protokol (většinou `http`) jako hodnotu a přidat ikonu, která zvýrazní všechny externí odkazy.

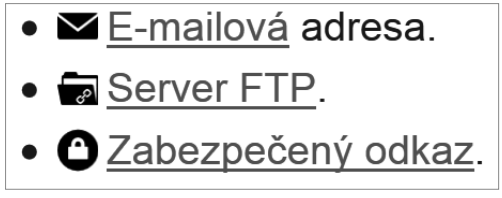
```
a[href^='http'] {
  background: url('odkaz.svg') no-repeat left center;
  display: inline-block;
  padding-left: 20px;
}
```

Výsledek můžete vidět na obrázku 3.1.

Obrázek 3.1 Ikona doplněna počátečním selektorem

Výše uvedený kód můžeme rozšířit tak, aby pokrýval i jiné protokoly, přičemž obrázek 3.2 ukazuje několik dalších protokolů (`mailto`, `ftp` a `https`) z následujícího příkladu.

```
a[href^='mailto'] { background-image: url('email.svg'); }
a[href^='ftp'] { background-image: url('slozka.svg'); }
a[href^='https'] { background-image: url('zamek.svg'); }
```



Obrázek 3.2 Další příklad tvorby ikon odkazů s použitím počátečního selektoru

Počáteční selektor se samozřejmě hodí především pro atributy, které mají rozsáhlejší hodnoty – například pro atribut `alt`, `cite` nebo `title`. Specifikace HTML5 přinesla spoustu nových formulářových elementů a atributů, a proto se stává tento selektor (a jemu příbuzné selektory) ještě užitečnější.

Ukažme si kupříkladu navržený atribut `datetime`, jenž přijímá datum – například `2015-03-14`:

```
<time datetime="2015-03-14">14. března<>
```

S počátečním selektorem bychom mohli aplikovat kaskádové styly na všechny elementy s daným rokem, což může být užitečné pro aplikace typu kalendář nebo archiv:

```
[datetime^='2015'] {...}
```

Koncový selektor hodnoty atributu

Koncový selektor funguje podobně jako počáteční selektor, ale samozřejmě z druhé strany. Hledáme tudíž elementy s atributem, jehož hodnota končí zadaným textem. Syntaxe se liší jen v jediném znaku. Tentokrát použijeme znak dolaru (`$`) před rovnítkem (`=`). Kompletní zápis vypadá takto:

```
E[atr$='hodnota'] {...}
```

Vraťme se opět k našemu ukázkovému kódu jazyka HTML z předchozí části, ale tentokrát uplatníme koncový selektor s novou hodnotou:

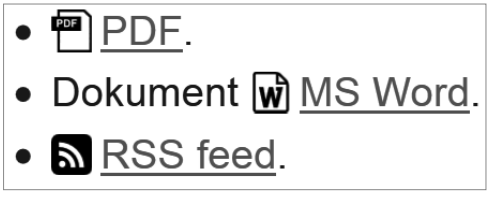
```
a[title$='knihovna'] {...}
```

V tomto případě aplikujeme dané pravidlo stylu na všechny položky seznamu, jelikož všechny hodnoty atributů `title` končí slovem `knihovna`.

Tento selektor můžeme použít (podobně jako počáteční selektor) k úpravě vzhledu hypertextových odkazů. Tentokrát se však nebudeme orientovat podle protokolů na začátku atributu `href`, ale podle přípon souborů na konci. Níže uvedený zdrojový kód obsahuje pravidla stylů pro spoustu oblíbených přípon souborů:

```
a[href$='.pdf'] { background-image: url('pdf.svg'); }
a[href$='.doc'] { background-image: url('word.svg'); }
a[href$='.rss'] { background-image: url('feed.svg'); }
```

Obrázek 3.3 ukazuje, jak by mohl vypadat výsledek.



Obrázek 3.3 Ikony u odkazů zobrazené pomocí koncových selektorů

Kdybychom chtěli dosáhnout stejného výsledku ve specifikaci CSS2, museli bychom do dokumentu přidat třídu (například `class="pdf"`). Koncové selektory mají tu výhodu, že s nimi lze detekovat odkazy na soubory automaticky, aniž bychom museli přidávat třídu. Nevýhodou tohoto řešení je, že někdy se přípony souborů nenacházejí na konci adres URI. Následující selektor nám ale umožňuje vypořádat se s touto situací.

Selektor libovolného podřetězce hodnoty atributu

Poslední nový atributový selektor budeme označovat jako **selektor libovolného podřetězce**. Funguje stejně jako předchozí dva selektory, ale hledá zadaný podřetězec kdekoliv v hodnotě atributu. Používá hvězdičku (`*`) před rovnítkem. Zde je syntaxe:

```
E[atr*='hodnota'] {...}
```

Pro demonstraci tohoto selektoru použijeme stejný kód jazyka HTML jako u počátečního a koncového selektoru. Náš selektor libovolného podřetězce bude vypadat následovně:

```
a[title*='obrazová'] {...}
```

Toto pravidlo stylu uplatňujeme u první a druhé položky našeho seznamu, protože obě obsahují slovo `obrazová` v atributu `title`, třebaže na jiné pozici v hodnotě.

Možná jste si všimli, že tento selektor se podobá částečnému selektoru hodnoty atributu ze specifikace CSS2. V tomto případě jsou dokonce zaměnitelné:

```
a[title~='obrazová'] {...}
```

Oba selektory se však výrazně liší. Kdybychom v našem kódu jazyka HTML použili specifikaci CSS3, mohli bychom vybrat stejné elementy jen pomocí tohoto krátkého podřetězce:

```
a[title*='ob'] {...}
```

Částečný selektor hodnoty atributu ale vyžaduje, abychom zadali podřetězec, jenž odpovídá celému slovu v seznamu slov oddělených mezerami – v tomto případě by se jednalo o slova `volně`, `dostupná`, `obrazová` a `knihovna`, takže selektor ze specifikace CSS2 by nenašel hodnotu `ob`.

Budeme pokračovat v příkladech, které jsme použili pro první dva atributové selektory – selektor libovolného podřetězce je rovněž možné použít pro přidávání ikon k souborům, jejichž adresa URI končí určitými příponami. Zamysleme se kupříkladu nad takovouto běžnou adresou URI:

```
<a href="http://www.priklad.cz/priklad.pdf?foo=bar">Příklad</a>
```

Kdybychom použili koncový selektor s hodnotou `pdf`, tento element by nebyl jeho platným cílem, přestože se jedná o soubor PDF, a to proto, že tato hodnota se nenachází až na konci dané adresy. Problém vyřešíme tak, že stejnou hodnotu použijeme v selektoru libovolného podřetězce. Tento atribut `href` totiž obsahuje podřetězec `.pdf`, a proto náš odkaz získá ikonu.

```
a[href*='.pdf'] { background-image: url('pdf.svg'); }
```

Tento selektor je nejflexibilnější ze všech tří nových atributových selektorů, protože umí vyhledávat podřetězce bez ohledu na to, kde se nacházejí. Zvýšená flexibilita s sebou přináší však i to, že musíme dávat větší pozor, když definujeme hodnoty pro tento selektor. Jednoduché kombinace písmen se můžou objevit kdekoliv v textovém řetězci, a proto ve výše uvedeném příkladu používáme raději hodnotu `.pdf` (příponu souboru), a ne hodnotu `pdf` (obvyklou zkratku).

Vícenásobné selektory atributů

Můžeme také řetězit více selektorů dohromady, díky čemuž dosáhneme slušné úrovně specifičnosti. Pomocí vícenásobných selektorů můžeme aplikovat pravidla stylů na atributy

s hodnotou na začátku, na konci nebo někde uprostřed. Představme si kupříkladu, že bychom měli dva soubory se stejnými názvy, ale umístěné v různých adresářích:

```
<p><a href="http://priklad.cz/adresar1/soubor.pdf">Příklad</a></p>
<p><a href="http://priklad.cz/adresar2/soubor.pdf">Příklad</a></p>
```

Kdybychom chtěli aplikovat pravidlo stylu jen na druhý element `p`, mohli bychom zřetězit několik selektorů:

```
a[href^='http://'][href*='/adresar2/'][href$='.pdf'] {...}
```

Prostřednictvím tohoto selektoru hledáme elementy, jejichž hodnota atributu `href` začíná předponou `http://`, končí příponou `.pdf` a obsahuje podřetězec `/adresar2/`. To je hodně specifické.

Kombinátor obecného sourozence

Posledním novým selektorem jazyka CSS3 je kombinátor. Jak už víme, kombinátor spojuje více selektorů. Kombinátor obecného sourozence rozšiřuje kombinátor sousedního sourozence, který byl zaveden ve specifikaci CSS2. Syntaxe se liší jen v jediném znaku:

```
E + F {...} /* Kombinátor sousedního sourozence */
E ~ F {...} /* Kombinátor obecného sourozence */
```

Rozdíl mezi nimi je nepatrný, ale důležitý – kombinátor sousedního sourozence vybírá element (*F*), jemuž bezprostředně předchází element (*E*) na stejné úrovni ve stromu dokumentu, ale kombinátor obecného sourozence vybírá elementy (*F*), kterým předchází element (*E*) na stejné úrovni ve stromu dokumentu, aniž by s nimi tento element musel bezprostředně sousedit.

Pokud se vám to zdá zmatené, prohlédněte si následující příklad. Zde jsou kaskádové styly:

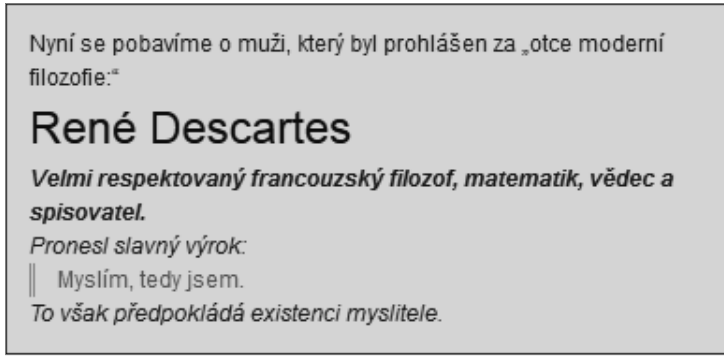
```
h2 + p { font-weight: bolder; } /* sousední sourozenec */
h2 ~ p { font-style: italic; } /* obecný sourozenec */
```

Aplikujete je na níže uvedený dokument (jenž je pro jednoduchost zkrácený):

```
<p>Nyní se pobavíme...</p>
<h2>René Descartes</h2>
<p>Velmi respektovaný francouzský filozof...</p>
<p>Pronesl slavný výrok:</p>
<blockquote>
  <p>Myslím, tedy jsem.</p>
</blockquote>
<p>To však předpokládá existenci myslitele.</p>
```

Výsledek můžete vidět na obrázku 3.4. V kódu jazyka CSS používáte kombinátor sousedního sourozence, abyste elementu `p`, který následuje bezprostředně za elementem `h2`, přidělili

tučné písmo. Navíc pomocí kombinátoru obecného sourozence nastavujete všem elementům `p` za elementem `h2`, které jsou na stejné úrovni, kurzívu.



Obrázek 3.4 Rozdíl mezi kombinátory sousedního sourozence a obecného sourozence

Odstavce `<p>Nyní se pobavíme...</p>` a `<p>Myslím, tedy jsem.</p>` nemají tučné písmo ani kurzívu. Jak je to možné? Protože první z nich se nachází před nadpisem `h2` a druhý se nachází uvnitř elementu `blockquote`, a tudíž je na jiné (nižší) úrovni ve stromu dokumentu. Proto nejsou ovlivněné našimi pravidly stylů.

Kdybychom chtěli nastavit kurzívu všem elementům `p` na stejné úrovni jako element `h2` bez použití kombinátoru obecného sourozence, museli bychom přidělit kurzívu všem elementům `p` a přidat samostatné pravidlo pro element `p` uvnitř elementu `blockquote`:

```
p { font-style: italic; }
blockquote p { font-style: normal; }
```

Kombinátor obecného sourozence nebudete pravděpodobně používat příliš často, protože se překrývá se spoustou jiných selektorů. Určitě ale narazíte na několik situací, kdy vám tento kombinátor může ušetřit trochu kódu (a času).

Shrnutí

Třebaže atributy jsou klíčovou součástí jazyka HTML4, většina z nich přijímá jen omezenou sadu hodnot, takže obvykle ani není nutné používat selektory atributů představené v této kapitole. Kromě atributu `href` má jen několik dalších atributů složitější hodnoty (například atributy `alt`, `class`, `id`, `rel` a `title`). Jazyk HTML5 zavádí atributy, jako jsou `datetime` nebo `pubdate`, s nimiž přichází i možnost kreativnějšího využití atributových selektorů.

Nové selektory z této kapitoly spolu se selektory z dřívějších verzí specifikace jazyka CSS umožňují aplikovat pravidla stylů podle specifikovaných elementů a atributů. Někdy ale nemusí být stylování na základě elementů a atributů dostačující. V takových případech

musíme přidat třídy nebo nesémantické elementy, které budou fungovat jako záchytné body pro naše styly. V kapitole 4 zjistíme, jak jazyk CSS3 odstraňuje tuto potřebu.

Selektory – podpora v prohlížečích

	Chrome	**Firefox**	**Safari**	**IE**
Nové atributové selektory	Ano	Ano	Ano	Ano
Kombinátor obecného sourozence	Ano	Ano	Ano	Ano

KAPITOLA 4

Pseudotřídy a pseudo-elementy

V této kapitole:

- Strukturní pseudotřídy
- Ostatní pseudotřídy
- Pseudoelementy
- Shrnutí
- Selektory atributů, pseudotřídy a pseudoelementy – podpora v prohlížečích

Úplně první specifikace jazyka CSS, specifikace CSS1, zavedla koncepci **pseudotříd** a **pseudoelementů**. Jedná se o selektory, které pracují s informacemi, jež rozšiřují (nebo se nacházejí mimo) strom dokumentu. Pseudotřídy rozlišují mezi různými stavy a typy elementu. Patří k nim například selektory, které poskytují informaci o stavu odkazu – `:hover`, `:visited`, `:active` atd. Pseudoelementy zprostředkovávají přístup k části elementu. Řadí se mezi ně kupříkladu pseudoelementy, které vybírají části textových uzlů – `:first-line` a `:first-letter`.

Výše zmíněné selektory jsou k dispozici už od první specifikace jazyka CSS, ale ve specifikaci CSS2.1 přibyla spousta dalších – třebaže výrobci prohlížečů implementovali pseudoelementy pořádně až poměrně nedávno. Specifikace CSS3 staví na těchto základech, přičemž rozšiřuje nabídku pseudotříd a také mírně upravuje syntaxi, aby je odlišila od pseudoelementů.

Výhoda existence více metod pro procházení dokumentu by měla být zřejmá – není nutné zavádět tolik záchytných bodů pro stylování. Pravděpodobně vám bude připadat následující kód povědomý:

```
<ul>
  <li class="prvni lichy"><span>L</span>orem ipsum</li>
  <li>Lorem ipsum</li>
  <li class="lichy">Lorem ipsum</li>
  <li class="posledni">Lorem ipsum</li>
</ul>
```

Předchozí kód obsahuje třídy, které popisují pozici jednotlivých elementů v dokumentu. Třídy `prvni` a `posledni` označují elementy `li`, jež jsou prvním a posledním dceřiným elementem elementu `ul`. Třídu `lichy` používáme u všech lichých elementů `li`. Navíc obalujeme dodatečný element `span` okolo prvního písmena v prvním elementu `li`.

Takový kód píšete v případě, že chcete nastylovat střídavé elementy, první nebo poslední element nebo jinak naformátovat první písmeno textového uzlu. Tento zápis však ubírá na čitelnosti a sémantice vašeho kódu, ale v mnoha případech se bez něj neobejdete, pokud chcete aplikovat některé kaskádové styly.

Nové metody jazyka CSS3 nám umožňují dosáhnout stejných výsledků, aniž bychom museli přidávat do dokumentu HTML zbytečné třídy a nesémantické elementy, díky čemuž vznikne čistější a snadněji udržovatelný kód:

```
<ul>
  <li>Lorem ipsum</li>
  <li>Lorem ipsum</li>
  <li>Lorem ipsum</li>
  <li>Lorem ipsum</li>
</ul>
```

Další přínos nových selektorů spočívá v tom, že když přidáme do dokumentu HTML nové elementy, nemusíme měnit třídy, abychom zachovali správné uspořádání. Tato změna posouvá jazyk CSS blíže k jím prosazovanému cíli – oddělit obsah od vzhledu.

Strukturní pseudotřídy

Z úvodu k této kapitole víme, že pseudotřídy umožňují vybírat elementy na základě informací, které nejsou uvedeny ve stromu dokumentu. K dispozici je celá řada různých podtypů, přičemž nejrozšířenější jsou tzv. **strukturní pseudotřídy**. Pomocí těchto podtypů hledáme elementy, jež nejsou dostupné prostřednictvím jednoduchých selektorů.

Vezměme si kupříkladu tento kód:

```
<div>
  <p>Lorem ipsum.</p>
  <p>Dolor sit amet.</p>
</div>
```

První ze dvou elementů `p` je prvním dceřiným elementem elementu `div`. To lze snadno vyčíst z dokumentu, ale strom dokumentu neobsahuje žádné informace, které by nám umožnily aplikovat pravidlo stylu jen na tento element. Z tohoto důvodu zavedla specifikace CSS2 pseudotřídu `:first-child`:

```
E:first-child {...}
```

Tato pseudotřída nám umožňuje vybrat element na základě informace, kterou neposkytuje žádný jeho atribut – to je přesný účel pseudotřídy. Protože pseudotřída `:first-child` byla zavedena již ve specifikaci CSS2, jednalo se o dosud jedinou pseudotřídu svého druhu. Specifikace CSS3 ale značně rozšířila možnosti, jelikož zavedla 11 nových strukturních pseudotříd.

Pseudotřídy nth-*

Čtyři nové pseudotřídy se zakládají na číselné hodnotě, jež slouží pro hledání pozice elementu ve stromu dokumentu. K zadání tohoto čísla používáme syntaxi `:nth-*`, přičemž hvězdička v tomto případě zastupuje množství nejrůznějších hodnot, které si představíme ve zbytku této kapitoly.

Základní syntaxe pseudotříd `:nth-*` je velmi jednoduchá. Standardně `n` reprezentuje číslo, které počítáme od 0 a zvyšujeme o 1 (1, 2, 3 atd.). Další celé číslo vkládáme jako násobek. Například `2n` označuje násobky čísla 2 (2, 4, 6 atd.), `3n` jsou násobky čísla 3 (3, 6, 9 atd.) atd.:

```
E:nth-*(n) {...}
E:nth-*(2n) {...}
E:nth-*(3n) {...}
```

Na prvním řádku používáme výchozí hodnotu `n`, takže vybereme všechny elementy *E*; výsledek bude stejný, jako bychom použili jednoduchý selektor elementu. Selektorem na druhém řádku hledáme každý druhý element *E* a posledním pak každý třetí element *E*.

Můžeme rovněž používat matematické operátory plus (+) a minus (-). Například pomocí pseudotřídy s `2n+1` vybíráme násobky dvou zvětšené o jedna (1, 3, 5 atd.) a prostřednictvím pseudotřídy s `3n+1` vyhledáme násobky tří zmenšené o jedna (2, 5, 8 atd.):

```
E:nth-*(n+1) {...}
E:nth-*(2n+1) {...}
E:nth-*(3n-1) {...}
```

Na prvním řádku vybíráme všechny elementy *E* s výjimkou prvního; tj. elementy s pořadím 2, 3, 4, 5 atd. Druhým selektorem hledáme všechny liché elementy *E* (1, 3, 5 atd.). Posledním řádkem vybíráme posloupnost elementů 2, 5, 8 atd.

K dispozici jsou také klíčová slova `even` (sudé) a `odd` (liché), s nimiž lze nahradit hodnoty `2n` a `2n+1`:

```
E:nth-*(even) {...}
E:nth-*(odd) {...}
```

Jako hodnotu můžeme rovněž použít `0n` (s nulou). Sama o sobě nemá žádné využití, ale je užitečná ve spojení s matematickým operátorem, jelikož můžeme ukázat na konkrétní element. Tento zápis je možné zkrátit tak, že uvedeme jen číslo za matematickým operátorem. Kdybychom chtěli vybrat kupříkladu třetí element, mohli bychom použít jakýkoliv z těchto selektorů:

```
E:nth-*(0n+3) {...}
E:nth-*(3) {...}
```

Jelikož jsme si popsali základní syntaxi, přesuňme se k samotným pseudotřídám.

Pseudotřídy :nth-child() a :nth-of-type()

Většina nových strukturních pseudotříd umožňuje vybírat elementy podle jejich pozice ve stromu dokumentu, a to vzhledem k jejich rodičovskému elementu (`-child`), nebo jejich klasifikace (`-of-type`). Někdy se překrývají, ale jsou mezi nimi podstatné rozdíly.

Nejjednoduššími příklady těchto pseudotříd jsou pseudotřídy `:nth-child()` a `:nth-of-type()`. Pseudotřída `:nth-child()` vybírá element na základě jeho pozice mezi všemi dceřinými elementy uvnitř jejich společného rodičovského elementu. Pseudotřída `:nth-of-type()` nebere v potaz všechny dceřiné elementy, ale pouze ty, které jsou daného typu.

```
E:nth-child(n) {...}
E:nth-of-type(n) {...}
E:nth-child(2n) {...}
E:nth-of-type(2n) {...}
```

První dvě pravidla stylů se shodují, protože jsme ponechali výchozí číselnou hodnotu (`n`). Obě uplatňujeme u všech dceřiných elementů typu *E*. Rozdíly se začínají vynořovat až u zbylých příkladů. Selektorem `nth-child(2n)` vybíráme elementy *E* mezi všemi sourozenci, ale jen ty, které jsou na sudých pozicích. Selektorem `nth-of-type(2n)` vybíráme všechny sudé elementy *E*, ale pouze mezi elementy tohoto typu.

Fungování těchto pseudotříd lze snadněji pochopit z praktických ukázek. Proto si ukážeme rozdíly na následujícím příkladu (který je pro zachování jednoduchosti zkrácený):

```
<div>
  <h2>The Picture of Dorian Gray</h2>
  <p>The artist is the creator...</p>
  <p>To reveal art and conceal the artist...</p>
  <p>The critic is he who can translate...</p>
</div>
```

A do naší šablony stylů vložíme tato dvě pravidla stylů:

```
p:nth-child(2n) { font-weight: bolder; }
p:nth-of-type(2n) { font-weight: bolder; }
```

Rozdílné výsledky po uplatnění obou těchto pravidel stylů lze vidět na obrázku 4.1. V našem ukázkovém dokumentu má element `div` čtyři dceřiné elementy: element `h2` a tři elementy `p`. Selektorem `n:nth-child(2n)` v prvním pravidle stylu hledáme každý druhý dceřiný element `p` (v tomto případě první a třetí odstavec) a nastavujeme mu tučné písmo, jak ukazuje obrazovka vlevo. Naproti tomu v druhém pravidle stylu máme selektor `p:nth-of-type(2n)`, jenž ignoruje element `h2`, a tudíž aplikujeme tučné písmo na každou druhou instanci elementu `p` mezi pouhými třemi elementy tohoto typu – tj. v našem případě jen na druhý element `p`.

Obrázek 4.1 Srovnání selektoru :nth-child() (vlevo) se selektorem :nth-of-type (vpravo)

Jak už víme a jak si lze odvodit s předchozích příkladů, selektory `:nth-child()` a `:nth-of-type()` se poměrně překrývají a můžeme je zaměňovat, což uděláme v následujícím příkladu.

Tabulka zobrazená na obrázku 4.2 vlevo ukazuje pětidenní předpověď počasí pro Londýn (proto jsou teploty uvedené ve stupních Celsia – 0 °C odpovídá 32 °F). Tyto údaje byly posbírány v lednu – ne vždy je tam taková zima. V tabulce najdeme všechny informace, které potřebujeme, ale bez rozlišení řádků lze tuto tabulku obtížně číst.

Nyní porovnejme tuto tabulku s tabulkou vpravo na stejném obrázku 4.2. U ní používáme techniku tvorby **pruhovaných tabulek**, abychom vizuálně rozdělili data, a ta se tak stala lépe čitelná.

Den	Počasí	Max. den (°C)	Min. noc (°C)	Vítr (km/h)
Ne	Slunečno	8	4	13
Po	Zataženo	7	4	10
Út	Zataženo	5	2	21
St	Sněžení	2	1	13
Čt	Déšť	4	2	11

Den	Počasí	Max. den (°C)	Min. noc (°C)	Vítr (m/h)
Ne	Slunečno	8	4	13
Po	Zataženo	7	4	10
Út	Zataženo	5	2	21
St	Sněžení	2	1	13
Čt	Déšť	4	2	11

Obrázek 4.2 Tabulka s předpovědí počasí (vlevo) a její lépe naformátovaná verze (vpravo). Údaje o počasí pocházejí ze serveru *http://bbc.co.uk/weather/*

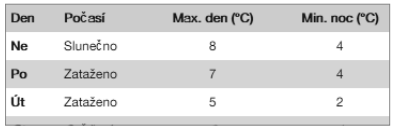
Toho výsledku jsme dosáhli pomocí jediného pravidla stylu jazyka CSS3:

```
tbody tr:nth-of-type(even) { background-color: #DDD; }
```

V tomto příkladu jsme mohli použít pseudotřídu `:nth-child()`, jelikož element `tbody` obsahuje pouze elementy stejného typu – `tr`. Pokud vybíráme mezi elementy stejného typu můžeme zaměňovat pseudotřídy `:nth-child()` a `:nth-of-type()`.

Pseudotřídy :nth-last-child() a :nth-last-of-type()

Pseudotřídy `:nth-last-child()` a `:nth-last-of-type()` přijímají stejné argumenty jako pseudotřídy `:nth-child()` a `:nth-of-type()`, až na to, že začínají počítat od posledního elementu – fungují tudíž obráceně. Řekněme kupříkladu, že budeme chtít v naší tabulce s počasím nějak naznačit, že předpovědi na čtyři a pět dní jsou méně přesné než na bližší dny. Na obrázku 4.3 je možné vidět, jak by to mělo vypadat.



Den	Počasí	Max. den (°C)	Min. noc (°C)
Ne	Slunečno	8	4
Po	Zataženo	7	4
Út	Zataženo	5	2

Obrázek 4.3 Dodatečné formátování pomocí pseudotřídy :nth-last-child()

Údaje na posledních dvou řádcích vypisujeme kurzívou prostřednictvím pseudotřídy `:nth-last-child()` (ačkoliv pseudotřída `:nth-last-of-type()` by v tomto případě odvedla stejnou práci), které jsme předali `-n+2` jako argument:

```
tbody tr:nth-last-child(-n+2) { font-style: italic; }
```

Použili jsme zápornou hodnotou (`-n`), abychom zvyšovali pořadí obráceně. Jelikož pseudotřídy `:nth-last-child()` a `:nth-last-of-type()` počítají pozpátku, pomocí záporné hodnoty začneme počítat dopředu. Pořadí začíná posledním elementem `tr` v tabulce a postupuje pozpátku, takže nejprve narazíme na poslední a předposlední řádky a nastavíme jim kurzívu.

Může se to zdát poněkud zmatené, ale rychle si na to zvyknete, až budete pravidelně procházet stromem dokumentu.

Pseudotřídy :first-of-type, :last-child a :last-of-type

Když si prohlédnete tabulky na obrázku 4.2, zjistíte, že text ve sloupci **Počasí** je zarovnaný doleva, zatímco následující sloupce jsou zarovnané na střed. Dosáhl jsem toho pomocí pseudotřídy `:first-of-type`, jež se podobá pseudotřídě `:first-child` ze specifikace CSS2 až na to, že vybírá jen elementy daného typu, jak už jste mohli vidět v této kapitole.

Jistě jste uhádli, že selektor `:first-child` vybírá element, který je prvním dceřiným elementem svého rodičovského elementu. Selektor `:first-of-type` je, podobně jako selektor `:nth-of-type()`, specifičtější – hledá jen element, jenž je prvním dceřiným elementem daného typu uvnitř svého rodičovského elementu. K dispozici jsou rovněž protějšky těchto pseudotříd – pseudotřídy `:last-child` a `:last-of-type`. Ty vybírají poslední dceřiný element a poslední dceřiný element daného typu.

V ukázkových tabulkách z předchozí části používáme níže uvedený kód pro jejich jednotlivé řádky:

```
<tr>
  <th>Ne</th>
  <td>Slunečno</td>
  <td>8</td>
  <td>4</td>
  <td>13</td>
</tr>
```