Louis Lazaris

sitepoint

CSS

OKAMŽITĚ

computer press

OVLÁDNĚTE CSS ZA VÍKEND

Louis Lazaris

CSS Okamžitě

Computer Press
Brno
2014

CSS Okamžitě

Louis Lazaris

Překlad: Ondřej Baše
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 978-80-251-4176-2

Vydalo nakladatelství Computer Press v Brně roku 2014 ve společnosti Albatros Media a. s.
se sídlem Na Pankráci 30, Praha 4. Číslo publikace 18 439.

1. vydání

ALBATROS MEDIA a.s.

Obsah

Poděkování

Chtěl bych poděkovat Simonu Mackiemu a Rachel Andrewové za vynikající rady a připomínky, které mi pomohly vylepšit tuto knihu.

O autorovi

Louis Lazaris je webový návrhář a bloger, který vytváří webové stránky již přes deset let. Můžete ho najít na sociální síti Twitter[1] nebo si přečíst více článků o jazyce CSS na jeho webových stránkách Impressive Webs[2].

1 *https://twitter.com/ImpressiveWebs*

2 *http://www.impressivewebs.com/*

Úvod

Pamatujete si na svou první vzdělávací hračku? Většina dětí se poprvé setkává se skládačkou s velkými díly, kterých bývá od 4 do 10 a které pasují do mezer na desce.

Jedná se o první zkušenost dítěte s porovnáváním tvarů dílků a mezer, což mu pomáhá rozvíjet jeho rozlišovací schopnosti.

Má manželka si myslí, že jsem pravděpodobně nikdy nedostal takovou skládačku. Vždy když uklízím nádobí, zaseknu se na plastových nádobách. Snažím se vkládat střední nádoby do menších a hranaté nádoby do kulatých. Manželka mi proto říká, že jsem jako dítě nikdy nezažil správný trénink. Ačkoliv předstírám, že si ze mě dělá legraci, tajně si uvědomuji, že má možná pravdu – skutečně už si své dětství tolik nevybavuji.

Pokud jste si koupili tuto malou knihu, ve světě jazyka CSS se podobáte malému dítěti s jeho první skládačkou. Doufám, že vás naučím o tomto jazyce co nejvíce a jak nejrychleji dokážu. Rovněž doufám, že se později ohlédnete a budete děkovat, že jste se „naučili skládat tvary dohromady".

Komu je tato kniha určena

Tato kniha je vhodná pro začínající webové návrháře a vývojáře. Předpokládá se základní znalost jazyka HTML.

Použité konvence

V této knize narazíte na řadu typografických konvencí a různých stylů rozvržení, které budou zvýrazňovat různé typy informací. Prohlédněte si proto následující konvence.

Ukázky zdrojového kódu

Zdrojový kód bude napsán neproporcionálním písmem; například takto:

```
<h1>Skvělý letní den</h1>
<p>Byl krásný letní den, který přímo vybízel k procházce parkem.
Ptáčci zpívali a všechny děti byly zpátky ve škole.</p>
```

Pokud se daný zdrojový kód nachází v archivu zdrojových kódů k této knize, příslušné jméno souboru se objeví nad tímto výpisem:

priklad.css

```
.zapati {
  background-color: #CCC;
  border-top: 1px solid #333;
}
```

Pokud bude výpis zobrazovat jen část obsahu souboru, bude označen slovem úryvek:

priklad.css *(úryvek)*

```
border-top: 1px solid #333;
```

Když budeme k současnému příkladu přidávat další zdrojový kód, takový kód se bude zobrazovat tučně:

```
function animuj() {
  VAR NOVA_PROMENNA = 'AHOJ';
}
```

Místo již napsaného zdrojového kódu bude výpis obvykle obsahovat tři tečky:

```
function animuj() {
  ...
  RETURN NOVA_PROMENNA;
}
```

Tipy, poznámky a upozornění

Tip: Haló, vy tam!
Tipy vám poskytují malé cenné rady.

Poznámka: Ehm, promiňte...
Poznámky jsou doplňující informace, které se týkají tématu, ale nejsou kriticky důležité.

Upozornění: Nezapomeňte vždy...
... věnovat pozornost těmto sdělením.

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/K2151* 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 nám můžete pomoci zlepšit následující vydání této knihy.

Veškerá existující errata zobrazíte na adrese *http://knihy.cpress.cz/K2151* po klepnutí na odkaz Soubory ke stažení.

KAPITOLA 1

Úvod do jazyka CSS

V této kapitole:
- Ukázkový projekt
- Jak stavět webové stránky
- Co je CSS
- Jak vložit styly do webové stránky
- Selektory jazyka CSS
- Používáme více selektorů
- Kaskáda a specifičnost
- Vždy používejte standardní režim
- Kostra ukázkové webové stránky

Vítejte do světa jazyka CSS. Tato kniha obsahuje základy jazyka CSS, díky nimž budete moct zkrášlovat své webové stránky, a to pomocí standardní technologie pro formátování webových stránek.

Tato kniha nebude povětšinou obsahovat strohé teoretické informace, ale především řadu praktických rad a příkladů. Ve velmi rychlém tempu si představíme nejrůznější stránky jazyka CSS a ukážeme si, jak s ním vytvářet webové stránky.

Ukázkový projekt

Abyste získali praktické zkušenosti s jazykem CSS, v této knize budete pracovat na ukázkovém projektu.

Cílem tohoto ukázkového projektu je vytvořit smyšlenou webovou stránku s názvem „Vyhledávač receptů". Kompletní verze je k dispozici na adrese *http://knihy.cpress.cz/K2151*.

Přejdeme od modelu k vývoji stránky. Návrh této stránky vznikl v programu Photoshop. Obrázek 1.1 ukazuje, jak by měla ukázková stránka vypadat.

Obrázek 1.1. Webová stránka, kterou budeme vytvářet v této knize

Jak stavět webové stránky

Zamysleme se, co přesně jazyk CSS je a jaký je jeho vztah k webovým stránkám. Webové stránky se zakládají na **obsahu**. Když navštívíme webovou stránku, zajímá nás obsah – text, fotografie, obrázky a videa. Obsah ukládáme prostřednictvím jazyka HTML, jejž si teď stručně shrneme.

Než začneme pracovat na našem projektu, seznámíme se s prvky jazyka CSS a způsoby, jak s nimi definovat vzhled našich webových stránek.

Jazyk HTML se skládá z **elementů**, a ty jsou obvykle tvořeny otevíracími a uzavíracími **značkami**. Těmito elementy sdělujeme webovým prohlížečům, jak by měly prezentovat obsah (text, fotografie atd.) uživateli. Například takto:

```
<header>
  <h1>Vyhledávač receptů</h1>
</header>
```

V tomto případě se na stránce zobrazuje text „Vyhledávač receptů“. Všechno ostatní, co se v předchozím kódu nachází (informace mezi ostrými závorkami), jsou značky jazyka HTML, které nejsou na stránce v prohlížeči viditelné. Tyto značky pouze označují část kódu, kde začíná a končí obsah. Z tohoto důvodu se jazyku HTML říká **značkovací jazyk** (HTML je zkratka pro anglický název Hyper Text Markup Language).

Tato kniha však není věnována jazyku HTML, pokud se chcete naučit více, zkuste kupříkladu některý z níže uvedených zdrojů:

- webové stránky „Jak psát web“[1],
- příručku HTML Concepts od nakladatelství SitePoint[2],
- článek „Introduction to HTML“[3] na webových stránkách Mozilla Developer Network.

Co je CSS

CSS je zkratka pro Cascading Style Sheets (česky **kaskádové styly**) – jedná se o samostatný, ale doplňující jazyk k jazyku HTML. S jazykem CSS aplikujeme styly na obsah našich webových stránek.

Nyní použijeme výše uvedený kód jazyka HTML, abychom se seznámili s jazykem CSS. Prozatím se nebudeme trápit přesným významem následujícího kódu, pouze si ukážeme, jak takový kód vypadá:

```
header {
  color: white;
  background-color: #333;
  font-size: 1.5em;
}
```

V předchozím kódu definujeme **pravidlo stylu**. Určitě by nemělo ujít naší pozornosti, že tři řádky kódu obalujeme složenými závorkami a že každý z těchto tří řádků obsahuje dvojtečku a středník. Vše uvnitř složených závorek nazýváme **deklarační blok**.

Částí, jež předchází první složené závorce, určujeme, jakou sekci webové stránky stylujeme. Této části říkáme **selektor**. Selektory si popíšeme podrobněji v příští kapitole. V tomto případě je cílem našeho pravidla stylu element `header` dokumentu HTML.

1 *http://www.jakpsatweb.cz/html/*

2 *http://reference.sitepoint.com/html/html-concepts*

3 *https://developer.mozilla.org/en-US/docs/HTML/Introduction*

Každý řádek deklaračního bloku označujeme (nikterak překvapivě) jako **deklaraci**. Každá deklarace se skládá z **vlastnosti** (část před dvojtečkou) a **hodnoty** (část za dvojtečkou). Kromě toho každá deklarace končí středníkem.

Právě jste viděli velmi jednoduchý příklad. Další příklady kódu jazyka CSS můžou být složitější, ale princip většiny z nich lze odhalit pouhým experimentováním, takže se nemusíte příliš obávat, když narazíte na něco, co neznáte.

Co tedy výše uvedený kód dělá? O konkrétních vlastnostech jazyka CSS se dozvíte později, avšak prozatím vydržte, než se seznámíte s důležitějšími základy, na kterých budete stavět po zbytek této knihy.

Jak vložit styly do webové stránky

Kaskádové styly můžeme vložit do webové stránky čtyřmi způsoby. Ty si nyní popíšeme, přičemž ten nejlepší si necháme na konec.

Vložené styly

Vzhled libovolného elementu na stránce je možné změnit prostřednictvím **vložených stylů**. Zde je příklad vloženého stylu u dříve uvedeného elementu dokumentu HTML:

```
<h1 style="color: blue; background-color: #333;">Vyhledávač receptů</h1>
```

V tomto případě jsme zapsali kód jazyka CSS do **atributu** s názvem `style`. Tímto atributem sdělujeme webovému prohlížeči, že uvnitř uvozovek bude následovat kód jazyka CSS. Takto aplikujeme styly jen na element, k němuž je připojujeme (tentokrát se jedná o element `h1`). Jedná se o neefektivní způsob vkládání kaskádových stylů, kterému bychom se měli vyhýbat za všech okolností.

Element style

Kód jazyka CSS můžeme vkládat do stránky taktéž s pomocí elementu style, jenž má kromě otevírací značky `<style>` také uzavírací značku `</style>`:

```
<style>
  header {
    color: white;
    background-color: #333;
    font-size: 1.5em;
  }
</style>
```

Nyní uplatňujeme kaskádové styly na element/y vybrané daným selektorem. Protentokrát tedy měníme vzhled všech elementů `header` na stránce, do které jsme vložili předchozí element `style`.

Používání direktivy @import uvnitř elementu style

Kód jazyka CSS lze rovněž zapisovat do samostatného souboru, který označujeme jako **šablona stylů**. Jedná se o prostý textový soubor s příponou *.css*. Externí kód jazyka CSS z takového souboru načteme do dokumentu HTML následujícím způsobem:

```
<style>
  @import url(css/styl.css);
</style>
```

Direktiva `@import` ale způsobuje problémy[4] – kdybychom načítali více šablon stylů prostřednictvím této direktivy, jejich stahování by trvalo déle než u následujícího způsobu. Z tohoto důvodu je lepší tuto direktivu nepoužívat.

Nejlepší způsob: element link

Kaskádové styly vložíme do stránky nejlépe pomocí elementu `link`:

```
<link rel="stylesheet" href="css/styl.css" />
```

Tento element, který umísťujeme do elementu `head` našeho dokumentu HTML, se v mnoha ohledech podobá direktivě `@import` – také slouží k načítání externího souboru. Díky tomu zachováme kód jazyka CSS oddělený od kódu jazyka HTML. Navíc tento postup načítání nevykazuje problémy jako direktiva `@import`. Protože kód jazyka CSS nevkládáme do více stránek (pouze se v nich na něho odkazujeme), lze tento kód snadněji udržovat (změnu nebo opravu stačí provést na jediném místě).

Selektory jazyka CSS

Už víme, že selektor jazyka CSS je součástí pravidla stylu a slouží k výběru obsahu, který chceme nastylovat. Ukažme si proto různé druhy dostupných selektorů i se stručnými popisy.

Univerzální selektor

Univerzální selektor funguje jako zástupný znak, s jehož pomocí můžeme vybrat všechny elementy na stránce. Na začátku této kapitoly jsme si řekli, že stránka HTML se skládá z obsahu

4 *http://www.stevesouders.com/blog/2009/04/09/dont-use-import/*

umístěného do značek. Jednotlivé skupiny značek reprezentují elementy. Nyní si předvedeme pravidlo stylu s univerzálním selektorem:

```
* {
  color: green;
  font-size: 20px;
  line-height: 25px;
}
```

Tři řádky kódu mezi složenými závorkami (tři deklarace vlastností `color`, `font-size` a `line-height`) aplikujeme na všechny elementy na stránce. Jak je patrné, univerzální selektor má podobu hvězdičky. Univerzální selektor však můžeme používat i v kombinaci s jinými selektory, jak zjistíme později v této kapitole.

Selektor typu elementu

Tento selektor se někdy zkráceně nazývá **selektor typu**, přičemž jak jeho název napovídá, slouží k vyhledávání elementů podle jejich jména. Selektorem `nav` bychom tedy vybrali všechny elementy `nav`, selektorem `ul` pak všechny neuspořádané seznamy (elementy `ul`) atd.

V níže uvedeném příkladu hledáme všechny elementy `ul` prostřednictvím selektoru typu:

```
ul {
  list-style: none;
  border: solid 1px #ccc;
}
```

Abychom si dali výše uvedený příklad do souvislosti, použijeme ho na níže uvedený kód jazyka HTML:

```
<ul>
  <li>Ryba</li>
  <li>Jablka</li>
  <li>Sýr</li>
</ul>

<div class="priklad">
  <p>Ukázkový text odstavce.</p>
</div>

<ul>
  <li>Voda</li>
  <li>Džus</li>
  <li>Javorový sirup</li>
</ul>
```

Tuto část stránky tvoří tři hlavní elementy – dva elementy `ul` a jeden element `div`. V tomto případě měníme vzhled obou elementů `ul`, ale nikoliv vzhled elementu `div`. Kdybychom jako selektor typu použili `div` místo `ul`, výše uvedené pravidlo stylu by platilo pouze pro element `div`, a ne pro elementy `ul`.

Povšimněte si, že předchozí styly nemají vliv na elementy uvnitř elementů `ul` a `div`. Nemusí tomu tak být vždy – vnitřní elementy totiž dědí některé styly, ale o tom se dozvíte více až později.

Selektor identifikátoru

Selektor identifikátoru uvozujeme mřížkou (znakem `#`), za níž následuje identifikátor. Samotný identifikátor je obyčejný textový řetězec, který si určí sám vývojář. Tento textový řetězec odpovídá hodnotě atributu `id` některého elementu na stránce.

Následuje příklad:

```
#kontejner {
  width: 960px;
  margin: 0 auto;
}
```

V předchozím kódu hledáme tento element prostřednictvím selektoru identifikátoru:

```
<div id="kontejner"></div>
```

V tomto případě nezáleží na tom, že jsme použili element `div` – mohlo by se jednat o jakýkoliv element jazyka HTML. Dokud mu necháme atribut `id` s hodnotou `kontejner`, prohlížeč na něho aplikuje naše pravidlo stylu.

Identifikátor elementu na webové stránce by měl být jedinečný. Jinými slovy – stránka by měla obsahovat pouze jeden element s identifikátorem `kontejner`. Kvůli tomu je také selektor identifikátoru málo flexibilní – pravidlo stylu, které ho obsahuje, lze totiž uplatnit jen jedenkrát na každé stránce.

Kdyby stránka obsahovala více elementů se stejným identifikátorem, kaskádové styly by stále fungovaly, ale z technického hlediska by taková stránka nebyla validním dokumentem jazyka HTML, což samozřejmě nechceme.

Kromě nízké flexibility trpí selektor identifikátoru rovněž problémem vysoké specifičnosti – tento problém si vysvětlíme dále v této kapitole.

Selektor třídy

Selektor třídy je pravděpodobně nejužitečnější selektor jazyka CSS. Tento selektor začíná tečkou, za kterou se nachází textový řetězec reprezentující název třídy. Vývojář si sám určuje název třídy, podobně jako u selektoru identifikátoru. Selektor třídy vybírá všechny elementy na stránce, jejichž hodnota atributu `class` obsahuje uvedenou třídu.

Mějme kupříkladu toto pravidlo stylu:

```
.panel {
  padding: 20px;
  margin: 10px;
  width: 240px;
}
```

To formátuje následující element jazyka HTML:

```
<div class="panel"></div>
```

Výše uvedené pravidlo stylu ale platí také pro jiné elementy s třídou `panel`. Mít na stránce více elementů se stejnou třídou je pro nás výhodné, protože můžeme opětovně používat pravidla stylů a vyhneme se zbytečnému opakování stejného kódu. Další výhodou selektoru třídy je, že je málo specifický – což si vysvětlíme později.

Selektor třídy je rovněž neocenitelným pomocníkem díky tomu, že jazyk HTML umožňuje přidělovat jedinému elementu více tříd. Jednoduše do atributu `class` zapíšeme několik tříd oddělených mezerami:

```
<div class="panel panel-dalsi panel-rozsireny"></div>
```

Kombinátor potomka

Selektor potomka (nebo přesněji **kombinátor potomka**) nám umožňuje kombinovat více selektorů, abychom mohli konkrétněji definovat náš výběr.

Například takto:

```
#kontejner .panel {
  float: left;
  padding-bottom: 15px;
}
```

Tento deklarační blok aplikujeme na všechny elementy s třídou `panel`, které se nacházejí uvnitř elementu s identifikátorem `kontejner`, a to s použitím kombinátoru potomka. Je důležité poznamenat, že element s třídou `panel` nemusí být přímo dceřiným elementem elementu s identifikátorem `kontejner` – bez problémů ho můžeme zabalit ještě do dalších elementů a naše pravidlo stylu bude stále fungovat.

Zde je ukázkový fragment dokumentu HTML:

```
<div id="kontejner">
  <div class="panel"></div>
  <div class="panel-2"></div>
</div>

<div class="panel"></div>
```

Kdybychom vložili naše pravidlo stylu do předchozí stránky, naformátovali bychom jen první element `div` s třídou `panel`. Element `div` s třídou `panel-2` by zůstal beze změny; stejně tak by se toto pravidlo stylu netýkalo druhého elementu `div` s třídou `panel`, jelikož ten se nachází vně elementu s identifikátorem `kontejner`.

Používejte kombinátor potomka ve svém kódu jazyka CSS obezřetně. Přestože tento typ selektoru vylepšuje čitelnost kódu, omezuje pravidla stylů na vybraný kontext – v tomto případě je kontextem element `div` s identifikátorem `kontejner` – pravidla stylů se proto stávají méně flexibilními.

Kombinátor dceřiného elementu

Selektor obsahující kombinátor dceřiného elementu se velmi podobá selektoru s kombinátorem potomka, ale vyhledává jen bezprostřední dceřiné elementy:

```
#kontejner > .panel {
  float: left;
  padding-bottom: 15px;
}
```

Tento kód se příliš neliší od příkladu kombinátoru potomka – místo samotné mezery používáme znak `>` (větší než; případně pravá ostrá závorka, chcete-li).

Pomocí tohoto selektoru hledáme všechny elementy, které mají třídu `panel` a jsou bezprostředními dceřinými elementy elementu s identifikátorem `kontejner`. Na rozdíl od kombinátoru potomka tedy nemůže být element s třídou `panel` obalený žádným jiným elementem, než je element s identifikátorem `kontejner`.

Ukázkový kód jazyka HTML by mohl vypadat například následovně:

```
<div id="kontejner">
  <div class="panel"></div>
  <div>
    <div class="panel"></div>
  </div>
</div>
```

Tentokrát uplatňujeme naše pravidlo stylu pouze na první element `div` s třídou `panel`. Druhý element `div` s třídou `panel` se totiž nachází uvnitř dalšího elementu `div`, a proto se k němu toto pravidlo stylu nevztahuje, ačkoliv má třídu `panel` a je potomkem elementu s identifikátorem `kontejner`.

Opět platí, že selektory s tímto kombinátorem můžou poněkud omezovat, ale také nám můžou pomoct – kupříkladu, když budeme chtít stylovat vnořené seznamy.

Kombinátor obecného sourozence

Selektor obsahující kombinátor obecného sourozence vybírá elementy na základě jejich sourozeneckých vztahů. To znamená, že vybrané elementy se nachází vedle sebe na stejné úrovni v kódu jazyka HTML.

```
h2 ~ p {
  margin-bottom: 20px;
}
```

V tomto typu selektoru používáme vlnovku (~). Předchozí pravidlo stylu uplatňujeme u elementů odstavců (`p`), ale jen u těch, které jsou sourozeneckými elementy alespoň jednoho elementu `h2`. Mezi elementem `h2` a odpovídajícím elementem `p` se může nacházet libovolný počet dalších elementů.

Použijme naše nové pravidlo stylu v následujícím dokumentu HTML:

```
<h2>Nadpis</h2>
<p>Ukázkový odstavec.</p>
<p>Ukázkový odstavec.</p>
<p>Ukázkový odstavec.</p>
<div class="panel">
  <p>Ukázkový odstavec.</p>
</div>
```

V tomto příkladu aplikujeme styly jen na první tři odstavce, protože poslední odstavec není sourozeneckým elementem elementu `h2` – nachází se totiž uvnitř elementu `div`.

Kombinátor sousedního sourozence

Selektor s kombinátorem sousedního sourozence poznáme podle znaménka plus (+) – jinak se téměř shoduje s kombinátorem obecného sourozence. Rozdíl spočívá v tom, že hledá bezprostředně nejbližší sourozenecký element, a ne jen sourozenecký element. Pravidlo stylu s tímto kombinátorem vypadá kupříkladu takto:

```
p + p {
  text-ident: 1.5em;
  margin-bottom: 0;
}
```

Výše uvedené pravidlo stylu uplatňujeme jen u odstavců, které bezprostředně následují za jiným odstavcem. To znamená, že pro první odstavec na stránce neplatí. Kdyby se mezi dvěma odstavci vyskytl jiný element, druhý z nich by také nebyl cílem tohoto pravidla stylu.

Když tedy použijeme tento selektor v níže uvedené stránce:

```
<h2>Nadpis</h2>
<p>Ukázkový odstavec.</p>
<p>Ukázkový odstavec.</p>
<p>Ukázkový odstavec.</p>

<div class="panel">
  <p>Ukázkový odstavec.</p>
  <p>Ukázkový odstavec.</p>
</div>
```

změníme vzhled jen druhého, třetího a pátého odstavce.

Selektor atributu

Selektorem atributu vybíráme elementy na základě hodnoty atributu, nebo dokonce přítomnosti samotného atributu, přičemž k tomu používáme hranatých závorek:

```
input[type="text"] {
  background-color: #444;
  width: 200px;
}
```

Před otevírací hranatou závorkou by neměla být mezera, pokud ovšem nezamýšlíte použít kombinátor potomka. Předchozím selektorem vyhledáte následující element:

```
<input type="text" />
```

Ale už ne tento element:

```
<input type="submit" />
```

Selektor atributu můžeme specifikovat rovněž bez hodnoty:

```
input[type] {
  background-color: #444;
  width: 200px;
}
```

Tentokrát vybíráme všechny elementy `input` s atributem `type`, a to bez ohledu na jeho hodnotu.

Selektor atributu je možné uvádět i bez názvu elementu vně hranatých závorek. V takovém případě vyhledáváme jakýkoliv element v závislosti na jeho atributu. Když zapisujeme hodnotu do selektoru atributu, můžeme ji vložit do uvozovek nebo apostrofů, ale také nemusíme (například hodnoty s mezerami musíme vkládat do uvozovek nebo apostrofů).

Pseudotřída

Pseudotřída označuje nějaký stav, v němž se element nachází. Před název pseudotřídy píšeme dvojtečku. Když kupříkladu uživatel přesune ukazatel myši nad element, tento element získá pseudotřídu `:hover`. Ukažme si příklad:

```
a:hover {
  color: red;
}
```

V tomto případě je součástí selektoru `a:hover` pseudotřída `:hover`. Jakmile uživatel přesune ukazatel myši nad libovolný odkaz na stránce (element `a`), změní se barva textu tohoto odkazu na červenou. Tato pseudotřída je dynamická, protože se objevuje a mizí v závislosti na uživatelské interakci – v tomto případě na přesunu ukazatele myši nad cílový element.

Neměla by nám uniknout důležitá skutečnost, že tímto typem selektoru nevybíráme pouze elementy, ale vybíráme elementy nacházející se v konkrétním stavu. V tomto příkladu jsme zvolili stav `hover`. Později si předvedeme také jiné pseudotřídy.

Pseudoelement

Jazyk CSS také nabízí typ selektoru, jejž nazýváme `pseudoelement`. Při vhodné aplikaci může být nesmírně užitečný. Tento selektor se však liší od jiných selektorů, s nimiž jsme se doposud setkali, proto si uvedeme rovnou praktický příklad:

```
.kontejner:before {
  content: "";
  display: block;
  width: 50px;
  height: 50px;
  background-color: #141414;
}
```

Nyní používáme pseudoelement `:before`. Z jeho názvu vyplývá, že vkládá do stránky imaginární element, a to do cílového elementu těsně před jeho obsah.

Na pseudoelementy se zaměříme důkladněji později v této knize.

Používáme více selektorů

Jednotlivé selektory, které jsme si právě popsali, můžeme kombinovat s libovolným počtem jiných selektorů. Ačkoliv je lepší se vyhýbat spojování mnoha selektorů dohromady, zde si uvedeme několik krátkých příkladů, abychom snadněji pronikli do této koncepce:

```
p.panel {
  color: blue;
}
```

Ve výše uvedeném zdrojovém kódu kombinujeme selektor typu elementu se selektorem třídy. V důsledku toho vybíráme jen ty odstavce, které mají třídu `panel`. Jedná se o špatně sestavený selektor, jemuž bychom se měli vyhýbat v téměř jakékoliv situaci. Mnohem lepší je použít pouze selektor třídy `.panel` a neomezovat ho selektorem typu elementu (v tomto případě `p`).

```
#formular [type=text] {
  border: solid 1px #ccc;
}
```

Tentokrát slučujeme selektor identifikátoru se selektorem atributu. Hledáme tak všechny elementy s hodnotou `text` v atributu `type`, které jsou umístěné uvnitř elementu s identifikátorem `formular`.

```
p, div, .panel {
  color: black;
}
```

V tomto případě oddělujeme selektory od sebe čárkami. Jedná se o užitečný způsob, jak uplatnit více selektorů v jediném pravidle stylu. Předchozí pravidlo stylu aplikujeme na všechny odstavce, všechny elementy `div` a všechny elementy s třídou `panel`.

Kaskáda a specifičnost

Na první pohled se mohou zdát následující koncepce obtížné. Pokud s jazykem CSS začínáte, existují metody, které vás mohou ušetřit spousty problémů, pokud se jimi budete řídit.

V prvé řadě si řekněme, co to znamená **kaskáda** a jak se k ní váže **specifičnost**.

Když webový prohlížeč provádí syntaktickou analýzu šablony stylů, postupuje odshora dolů, přičemž přednost dává deklaracím, které najde níže. Pokud tedy spolu dvě deklarace kolidují, později uvedená deklarace přepíše tu dřívější.

Prioritu a přepisování deklarací si můžeme ukázat na jednoduchém příkladu. Předpokládejme, že v jediném souboru s kódem jazyka CSS máme dvě následující pravidla stylů:

```
p {
  font-size: 20px;
}

p {
  font-size: 30px;
}
```

V obou pravidlech stylů používáme stejný selektor – obě platí pro všechny odstavce. Liší se však v hodnotě vlastnosti `font-size`. Hodnotu vlastnosti `font-size` tudíž definujeme dvakrát pro stejný element. Jakou velikost písma budou mít nakonec odstavce? Jelikož pravidlo stylu s velikostí písma `30px` následuje za prvním pravidlem stylu, odstavce budou mít velikost písma 30 pixelů.

Uvedli jsme si velmi jednoduchý příklad, který však demonstruje, že pozdější deklarace vlastností mají přednost před dřívějšími ve stejném souboru. Jednoduché, že? Bohužel se to trochu zkomplikuje, protože selektory mohou mít různé úrovně specifičnosti. Zde je příklad:

```
div p {
  color: blue;
}

p {
  color: red;
}
```

Kdybychom všechny elementy `p` na webové stránce vložili do nějakého elementu `div`, deklarace z prvního pravidla stylu (se selektorem `div p`) by přebily deklarace shodných vlastností z druhého pravidla stylu (se selektorem `p`). Je tomu tak proto, že selektor potomka (z prvního pravidla stylu) má přednost před selektorem typu elementu (z druhého pravidla stylu).

V tomto případě by tudíž všechny odstavce uvnitř elementů `div` měly modrý text, a to bez ohledu na to, že se druhé pravidlo stylu v šabloně stylů nachází až za prvním pravidlem stylu. Třebaže se prohlížeč řídí pořadím pravidel stylů, mnohem více ho zajímá specifičnost selektorů.

Zde je další příklad:

```
#hlavni {
  color: green;
}

body div.kontejner {
  color: pink;
}
```

Předpokládejme, že máme následující element:

```
<div id="hlavni" class="kontejner"></div>
```

Mohlo by se zdát, že druhé pravidlo stylu, které má dva selektory typu, selektor třídy a selektor potomka, bude mít přednost před prvním pravidlem stylu, v němž používáme pouze selektor identifikátoru. Není tomu tak. Náš element `div` bude mít zelený text, protože selektor identifikátoru má velmi vysokou specifičnost, a tudíž má přednost před složeným selektorem z druhého pravidla stylu.

Když se bavíme o přepisování vlastností a prioritě selektorů, máme na mysli pouze selektory z pravidel stylů, které cílí na stejné elementy a deklarují stejné vlastnosti. Kdybychom kupříkladu v prvním pravidle stylu se selektorem `#hlavni` specifikovali barvu textu, zatímco v druhém pravidle stylu se selektorem body `div.kontejner` bychom definovali jinou vlastnost, platily by obě deklarace vlastností a k žádnému přepsání by nedošlo. Stručně řečeno – specifičnost selektoru neovlivňuje všechny deklarace, ale jen duplicitní deklarace napříč různými pravidly stylů.

Teď se konečně ukázala pravá síla selektoru identifikátoru – málokterý selektor ho dokáže přebít, a proto byste se mu měli vyhýbat, jak je to jen možné. Vývojáři pracující v jazyce CSS občas vybízejí k použití selektoru identifikátoru pro jedinečné elementy a koneckonců tento selektor je platným selektorem dle standardu jazyka CSS. Zapamatujte si však, že nadměrné používání tohoto selektoru vás může přivést k celé řadě problémů, a to zejména ve větších projektech.

Abyste se nechytili do pasti se specifičností selektorů, používejte raději ve velké míře selektory tříd a naučte se zapojovat techniku zvanou **objektově orientované CSS** (nebo také **modulární CSS**). Více informací najdete například v článku „And Introduction To Object Oriented CSS“[5].

Poznámka: Rozdílné zacházení s identifikátory

Doporučení vyhýbat se selektorům identifikátorů platí jen pro jazyk CSS, nikoliv pro identifikátory v jazyce HTML. Atribut `id` je velmi důležitý pro místní odkazy na stránce (kotvy) a také pro přístup k elementům z jazyka JavaScript.

Jiné typy selektorů rovněž přidávají úroveň specifičnosti, ale už nezpůsobují tolik problémů. Nebojte se proto používat pseudotřídy nebo selektory potomků – je to naprosto v pořádku. Používejte je, kdykoliv je shledáte užitečnými.

Na Internetu najdete spoustu článků, které popisují kaskádu a specifičnost. Až získáte základní znalosti z této knihy, zkuste si prohlédnout kupříkladu příručku jazyka CSS od nakladatelství SitePoint[6].

Vždy používejte standardní režim

Základní poučka radí, abyste pro každý svůj dokument HTML používali **standardní režim**. Přestože může tato rada vyznívat poměrně technicky náročně, ve skutečnosti je velmi jednoduché zajistit, aby všechny vaše stránky jazyka HTML používaly tento režim.

Aby webový prohlížeč zobrazoval vaše kaskádové styly správně (to také dělá, pokud se nachází ve standardním režimu), ujistěte se, že úplně nahoře každého dokumentu HTML máte tuto značku:

```
<!DOCTYPE html>
```

5 *http://coding.smashingmagazine.com/2011/12/12/an-introduction-to-object-oriented-css-oocss/*

6 *http://reference.sitepoint.com/css/inheritancecascade*