Odkryj tajemnice CSS — projektuj stylowo!

CSS

nieoficjalny podręcznik

Wydanie IV

David Sawyer McFarland

O'REILLY®

Helion

Tytuł oryginału: CSS: The Missing Manual, 4th Edition

Tłumaczenie: Łukasz Piwko

ISBN: 978-83-283-2289-9

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/cssnp4
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/cssnp4.zip

Printed in Poland.

Spis treści

ROZDZIAŁ

3

Selektory, czyli do czego odnoszą się style

Każdy styl CSS składa się z dwóch podstawowych części: selektora i bloku deklaracji (pisałem już o tym w poprzednim rozdziale). Blok deklaracji zawiera właściwości formatujące kolor, rozmiar pisma itd., ale to służy tylko upiększaniu strony. Prawdziwa magia CSS kryje się w tych kilku pierwszych znakach poprzedzających każdą regułę — selektorze. Selektory, mówiąc arkuszom stylów, *co* mają sformatować, dają nam pełną kontrolę nad wyglądem strony (rysunek 3.1). Jeśli chcemy, selektor może odnosić się do wielu elementów strony jednocześnie lub, jeśli wolimy skupić się na szczegółach, do jednego określonego elementu albo jednej grupy elementów. Selektory CSS dają bardzo duże możliwości i w tym rozdziale nauczymy się je wykorzystywać.

```
h1 {
    font-family: Arial, sans-serif;
    color: #CCCCFF;
}
```

RYSUNEK 3.1.
Pierwsza część stylu, selektor, wskazuje, który element lub które elementy na stronie mają być sformatowane. W tym przypadku h1 oznacza, że formatowanie ma zostać zastosowane do wszystkich nagłówków pierwszego rzędu lub znaczników <h1> na tej stronie

UWAGA Osoby, które wolą zdobyć nieco doświadczenia, zanim zaczną czytać na temat zasad dotyczących selektorów CSS, powinny przejść do kursu w dalszej części rozdziału.

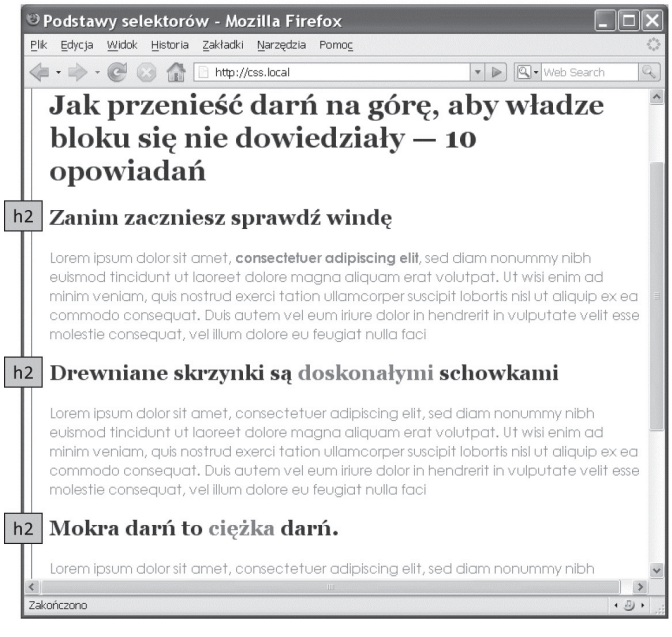
Selektory typu — style dla elementów HTML

Selektory służące do wskazywania określonych elementów HTML nazywają się **selektorami typu** (ang. *type selector*) lub **elementu** (ang. *element selector*). Są one niezwykle skuteczne, ponieważ mają zastosowanie do każdego wystąpienia danego znacznika na stronie. Dzięki nim można dokonać znacznych zmian w projekcie strony przy bardzo małym nakładzie pracy. Przykładowo aby sformatować wszystkie akapity tekstu na stronie, nadając im taki sam krój, kolor i rozmiar pisma, wystarczy utworzyć styl wykorzystujący selektor `p` (od znacznika `<p>`). W zasadzie selektor znacznika przedefiniowuje sposób prezentacji przez przeglądarkę zawartości określonego znacznika.

Przed powstaniem CSS tekst formatowano, umieszczając go w znacznikach `<font>`. Żeby sprawić, aby wszystkie akapity na stronie wyglądały tak samo, trzeba było użyć tego znacznika wiele razy. Wymagało to wiele pracy i powodowało powstawanie dużych ilości kodu HTML, co z kolei pociągało za sobą wydłużony czas pobierania i aktualizacji strony. Przy zastosowaniu selektorów znacznika poza kodem HTML nie trzeba w zasadzie robić nic — wystarczy utworzyć regułę CSS, a przeglądarka zajmie się resztą.

Selektory typu łatwo odnaleźć w regułach CSS, ponieważ ich nazwy pokrywają się z nazwami znaczników, do których się odwołują — `p`, `h1`, `table`, `img` itd. Na przykład na rysunku 3.2 selektor `h2` (na górze) nadaje styl pisma wszystkim znacznikom `<h2>` na stronie (na dole).

```
h2 {
   font-family:"Century Gothic", "Gill Sans", sans-serif;
   color:#000000
   margin-bottom:0;
}
```



RYSUNEK 3.2. *Selektor typu zmienia wygląd wszystkich wystąpień danego znacznika na stronie. Ta strona ma trzy znaczniki <h2> (oznaczone odpowiednio po lewej stronie okna przeglądarki). Pojedynczy styl z selektorem h2 pozwala kontrolować wygląd wszystkich znaczników <h2> na stronie*

UWAGA Jak wyraźnie widać na rysunku 3.2, w selektorach nie używa się znaków mniejszości (<) i większości (>), które otaczają znaczniki HTML. W związku z tym, pisząc regułę na przykład dla znacznika `<p>`, wystarczy podać tylko jego nazwę, czyli `p`.

Selektory typu mają jednak też wady. Co zrobić, jeśli chcemy, aby część akapitów wyglądała tak, a pozostałe inaczej? Prosty selektor znacznika nie sprawdzi się w takiej sytuacji, ponieważ nie dostarcza on przeglądarce wystarczających informacji, aby ta mogła odróżnić akapity, które mają być pisane fioletowym, powiększonym i pogrubionym pismem, od tych, które mają być pisane normalnym czarnym pismem. Na szczęście w CSS dostępnych jest kilka rozwiązań tego problemu, z których najprostsze polega na użyciu tzw. *selektora klasy*.

Selektor klasy — precyzyjna kontrola

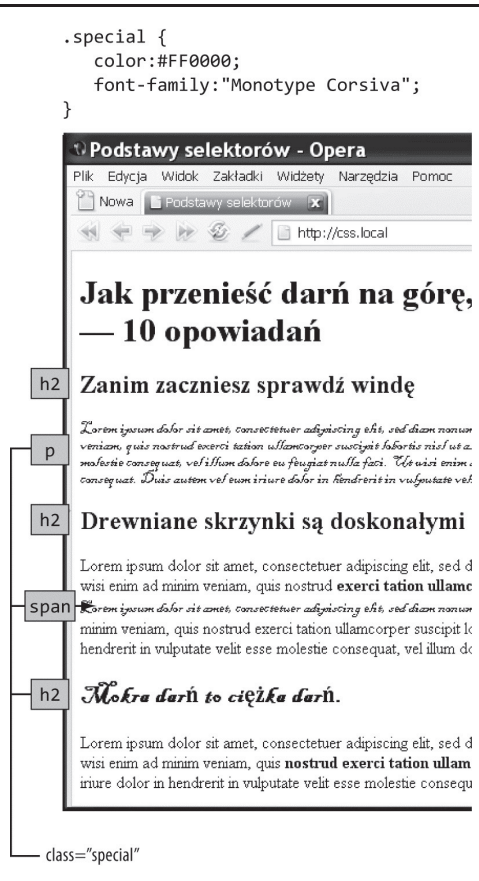
Gdy chcemy kilka wybranych elementów strony zdefiniować inaczej niż pozostałe elementy tego samego typu na tej stronie — na przykład chcemy dwóm obrazom nadać czerwone obramowanie, a pozostałe grafiki pozostawić bez zmian — możemy użyć selektora *klasy*. Jeżeli używałeś kiedyś stylów w edytorze tekstów typu Microsoft Word, to znasz już zasadę działania selektora klasy. Selektor taki tworzy się poprzez zdefiniowanie jego nazwy, a następnie stosuje się go do wybranych elementów na stronie. Można na przykład utworzyć styl o nazwie `.copyright` i stosować go tylko do tych akapitów, które zawierają informacje chronione prawami autorskimi, nie wpływając na wygląd pozostałych akapitów.

Selektory klas pozwalają także na odniesienie się do konkretnego elementu na stronie, bez względu na jego znacznik. Powiedzmy, że chcemy sformatować jedno lub dwa słowa, które znajdują się wewnątrz akapitu. Nie chcemy zmieniać wyglądu całego znacznika `<p>`, a tylko jakiegoś znajdującego się w nim wyrażenia. Do wskazania tych słów można wykorzystać selektor klasy. Za pomocą tego selektora można nawet zastosować takie samo formatowanie do różnych elementów, które mają różne znaczniki HTML. Przykładowo jednemu akapitowi i nagłówkowi drugiego rzędu można nadać ten sam styl — mogą to być na przykład ten sam kolor i krój pisma, za pomocą których wyróżniamy ważne informacje, co pokazano na rysunku 3.3. W przeciwieństwie do selektorów znaczników, które ograniczają się tylko do znaczników HTML występujących na stronie, selektory klas mogą występować w dowolnej liczbie i mogą znajdować się w dowolnym miejscu.

UWAGA Aby zastosować selektor klasy tylko do kilku słów zawartych w innym znaczniku (na przykład `<p>` z rysunku 3.3), trzeba skorzystać z pomocy znacznika `<span>`. Więcej szczegółów na ten temat znajduje się w ramce „Znaczniki i ”.

Jak nietrudno zauważyć, na początku każdego selektora klasy znajduje się kropka, na przykład `.copyright` czy `.special`. Jest to jedna z kilku reguł, o których należy pamiętać przy nadawaniu nazw klasom:

- **Nazwy wszystkich selektorów klas muszą zaczynać się od kropki**. Dzięki niej przeglądarka dowiaduje się o selektorach klas zawartych w arkuszach stylów.



RYSUNEK 3.3. *Selektory klas pozwalają na dokonywanie precyzyjnych zmian w projekcie. Można na przykład zmienić styl jednego nagłówka <h2> (Mokra darń to ciężka darń). Selektor klasy .special każe przeglądarce, aby zastosowała ten styl tylko do tego jednego znacznika <h2>. Klasy można stosować także do innych znaczników, jak górny akapit na tej stronie*

- **W nazwach klas można używać tylko liter, cyfr, łączników i znaków podkreślenia**.
- **Bezpośrednio po kropce zawsze musi znajdować się *litera***. Na przykład `.9lives` nie jest poprawną nazwą klasy, ale `.crazy8` tak. Nazwa klasy może wyglądać tak: `.copy-right` lub `.banner_image`, ale nie `.-zle` lub `._tez_zle`.
- **W nazwach klas rozróżniane są małe i wielkie litery**. Przykładowo `.SIDEBAR` i `.sidebar` to dwie różne klasy w CSS.

Poza nazwą proces tworzenia stylów klas nie różni się od stylów znaczników. Po nazwie klasy powinien znajdować się blok deklaracji zawierający wszystkie żądane style:

```
.special {
color:#FF0000;
font-family:"Monotype Corsiva";
}
```

Jako że selektory typu odnoszą się do wszystkich znaczników na stronie, wystarczy je zdefiniować w arkuszu stylów, a znaczniki HTML, na których one działają, są już na swoich miejscach. Wolność, jaką dają selektory klas, jest okupiona niedużym dodatkiem pracy. Używanie selektorów klas jest procesem dwufazowym. Po utworzeniu reguły klasy trzeba jeszcze wskazać, *do czego* ma być ona stosowana. W tym celu do znacznika HTML, któremu ma być nadany styl, dodaje się atrybut `class`.

Załóżmy, że utworzyliśmy klasę `.special`, za pomocą której będziemy wyróżniać określone elementy na stronie. Aby dodać ten styl do akapitu, należy do znacznika `<p>` dodać atrybut `class` w następujący sposób:

```
<p class-"special">
```

UWAGA W HTML-u przed nazwą klasy będącą wartością atrybutu `class` nie stawia się kropki. Jest ona wymagana tylko przed nazwą selektora w arkuszu stylów.

Kiedy przeglądarka napotyka ten znacznik, wie, że do akapitu należy zastosować reguły formatujące zawarte w stylu `.special`. Dzięki klasom można także sformatować tylko określony fragment akapitu bądź nagłówka, posługując się znacznikiem `<span>`. Na przykład aby wyróżnić tylko jedno słowo w akapicie przypisanym do klasy `.special`, można napisać taki kod:

```
<p>Witaj w <span class="companyName">Café Soylent Green</span>,
➥restauracji z duszą.</p>
```

Po utworzeniu stylu klasy można go zastosować do dowolnego znacznika na stronie. Jedną klasę można zastosować do dowolnej liczby znaczników, a więc można utworzyć specjalny styl (`.special`) o określonych właściwościach czcionki i koloru i zastosować go do elementów `<h2>`, `<p>`, `<ul>` itp.

Jeden element, wiele klas

Nie tylko jedną klasę można przypisać do wielu elementów, lecz również jednemu elementowi można przypisać wiele klas. Choć w pierwszej chwili może się wydawać, że tworzenie wielu klas to mnóstwo pracy, jest to bardzo często stosowana technika.

Oto przykład sytuacji, gdy dobrym rozwiązaniem jest przypisanie jednemu elementowi kilku klas: wyobraź sobie, że projektujesz interfejs do zarządzania koszykiem zakupów w sklepie internetowym. W interfejsie tym znajdują się różne przyciski, z których każdy służy do czegoś innego. Jeden na przykład umożliwia usunięcie produktu z koszyka, inny dodanie pozycji, a jeszcze inny służy do zmieniania liczby kupowanych produktów.

Dobry projektant wie, że wszystkie przyciski powinny mieć pewne cechy wspólne, np. zaokrąglone rogi i taki sam krój pisma, ale też powinny się od siebie wyraźnie różnić, np. przycisk usuwania jest czerwony, przycisk dodawania produktów jest zielony itd. Te podobieństwa i różnice można zdefiniować właśnie w kilku klasach. Jedna klasa będzie miała zastosowanie do wszystkich przycisków, a pozostałe — tylko do wybranych typów przycisków.

PRZYSPIESZAMY

Znaczniki i

W rozdziale 1. wprowadziłem dwa ogólne znaczniki HTML `<div>` i `<span>`, które można dopasować do własnych potrzeb za pomocą CSS. Jeśli nie ma znacznika HTML, który dokładnie odpowiada miejscu, w jakim chcemy umieścić styl klasy albo identyfikator (ID), to należy użyć znacznika `<span>` lub `<div>`.

Znacznik `<div>` identyfikuje logiczną *sekcję* strony, jak baner, pasek nawigacyjny, pasek boczny czy stopka. Można w nim także umieszczać inne elementy, które zawierają określone fragmenty strony, jak nagłówki, listy wypunktowane czy akapity (programiści nazywają je elementami *blokowymi*, ponieważ tworzą one pełne bloki treści otoczone nowymi wierszami). Znacznik `<div>` jest podobny w użyciu do znacznika `<p>` — wpisuje się otwierający znacznik `<div>`, wprowadza tekst, obrazy i inną treść, a następnie zamyka go znacznikiem `</div>`.

Znacznik `<div>` ma unikalną zdolność zawierania *wielu* elementów blokowych, dzięki czemu doskonale nadaje się do grupowania logicznie ze sobą powiązanych znaczników, takich jak logo i pasek nawigacyjny w banerze na stronie albo seria wiadomości na pasku bocznym. Mając pogrupowane w ten sposób elementy, formatowanie można stosować do każdego ze znaczników znajdujących się w `<div>` osobno lub przenieść cały blok treści znajdujący się w tym znaczniku w określone miejsce, na przykład na prawą stronę okna przeglądarki (CSS pozwala na kontrolowanie układu strony w taki sposób, ale o tym piszę dopiero w części trzeciej).

Wyobraźmy sobie na przykład, że dodaliśmy do strony obraz, któremu towarzyszy podpis. Za pomocą znacznika `<div>` (z klasą) można zgrupować to zdjęcie razem z podpisem:

```
<div class="photo">
<img src="holidays.jpg" alt="Pingwiny
➥zaczynają figlować"/>
<p>Mama, tata i ja na naszej corocznej
➥wycieczce na Antarktydę.</p>
</div>
```

W zależności od tego, co zostanie umieszczone w bloku deklaracji, klasa `.photo` może dodawać dekoracyjne obramowanie, kolor tła itd. zarówno do zdjęcia, jak i do podpisu pod nim. W części trzeciej tej książki prezentuję jeszcze bardziej przydatne zastosowanie znacznika `<div>`, włącznie z jego zagnieżdżaniem.

W najnowszej wersji języka HTML wprowadzono wiele elementów blokowych, które zachowują się podobnie jak elementy `<div>`, ale mają bardziej konkretne przeznaczenie. Na przykład obrazy z podpisami można prezentować za pomocą nowego elementu `<figure>`. Ponieważ jednak przeglądarka IE8 nie obsługuje tych nowych elementów (zobacz ramkę „Jak zmusić IE 8 do obsługi elementów HTML5” w rozdziale 1.), wielu projektantów nadal do grupowania używa tylko elementów `<div>`.

Ponadto nowe elementy HTML5 mają nadawać częściom stron „znaczenie”. Na przykład element `<article>` oznacza samodzielny blok tekstu, taki jak np. artykuł w czasopiśmie. Ale nie każdy kod HTML musi mieć określoną semantykę, więc mimo wszystko nadal często będziesz potrzebować elementu `<div>` do grupowania innych elementów.

Znacznik `<span>` natomiast pozwala zastosować style klasy lub identyfikatora tylko do określonego *fragmentu* jakiegoś znacznika. Znacznik `<span>` można stosować na pojedynczych słowach lub wyrażeniach (często nazywanych elementami *śródliniowymi*) wewnątrz akapitów w celu nadania im indywidualnych cech formatowania. W poniższym przykładzie klasa o nazwie `.companyName` nadaje styl elementom śródliniowym „CosmoFarmer.com”, „Disney” i „ESPN”.

```
<p>Witamy w <span class="companyName"
➥>CosmoFarmer.com</span>, firmie
➥rodzicielskiej takich znanych korporacji,
➥jak <span class="companyName">Disney</
➥span> i <span class="companyName">ESPN</
➥span>...no może nie do końca.</p>
```

Najpierw zdefiniujemy klasę ogólną o nazwie `.btn`:

```
.btn {
  border-radius: 5px;
  font-family: Arial, Helvetica, serif;
  font-size: .8 em;
}
```

Następnie dodamy klasy dla poszczególnych rodzajów przycisków:

```
.delete {
  background-color: red;
}
.add {
  background-color: green;
}
.edit {
  background-color: grey;
}
```

Teraz danemu elementowi można przypisać dowolną kombinację tych klas, aby otrzymać spójny zestaw różniących się od siebie detalami przycisków:

```
<button class="btn add">Dodaj</button>
<button class="btn delete">Usuń</button>
<button class="btn edit">Edytuj</button>
```

Przeglądarki internetowe doskonale radzą sobie ze stosowaniem deklaracji z wielu klas do pojedynczych elementów i jest to zgodne z zasadami języka HTML. W kodzie HTML wystarczy zdefiniować elementowi atrybut `class` i w jego wartości wpisać tyle nazw klas oddzielanych spacjami, ile się chce. Przeglądarka połączy deklaracje z wszystkich wskazanych reguł i zastosuje je jako jedną regułę. Zatem w naszym przykładzie wszystkie przyciski będą miały zaokrąglone rogi oraz pismo o rozmiarze `0.8em` i kroju Arial. Jednocześnie przycisk dodawania produktów będzie zielony, przycisk usuwania produktów będzie czerwony, a przycisk edycji — szary.

Zaletą tej metody jest to, że gdy uznamy, że przyciski nie powinny jednak mieć zaokrąglonych rogów albo powinny mieć inny krój pisma, wystarczy zmienić tylko styl `.btn`, który odnosi się do wszystkich przycisków. Analogicznie: jeśli postanowimy zmienić kolor przycisku edycji na żółty, to zmiana wprowadzona w stylu `.edit` znajdzie zastosowanie właśnie tylko w tym przycisku.

Selektor ID — pojedyncze elementy strony

Selektor identyfikatora w CSS służy do *identyfikacji* niepowtarzalnych części strony, takich jak banery, paski nawigacyjne czy główny obszar z treścią. Podobnie jak w przypadku selektora klasy, identyfikator tworzy się poprzez nadanie mu nazwy w CSS, a następnie stosuje się go, dodając do kodu HTML. Jaka jest zatem między nimi różnica? Jak napisano w ramce „Ukryte moce selektorów identyfikatora", selektor identyfikatora ma pewne określone zastosowania na bardzo długich lub wykorzystujących JavaScript stronach. Poza tym powodów do używania identyfikatorów zamiast klas jest niewiele.

UWAGA Ostatnio można zaobserwować trend odchodzenia od stosowania selektorów identyfikatora. Zrozumienie przyczyn tego zjawiska wymaga trochę większej wiedzy niż przedstawiona do tej pory. Dlatego też w książce tej nie zobaczysz zbyt wielu tych selektorów, a szczegółowe informacje na temat tego, dlaczego ich stosowanie nie jest najlepszym pomysłem, znajdują się w ramce „Ukryte moce selektorów identyfikatora" w dalszej części rozdziału.

Mimo że wielu twórców stron internetowych nie używa już identyfikatorów tak często jak kiedyś, warto jednak wiedzieć, czym one są i jak działają. Jeśli zdecydujesz się użyć selektora identyfikatora, to nie będziesz mieć z tym żadnego problemu, ponieważ jest to bardzo łatwe. Podobnie jak w klasach używa się kropki, w identyfikatorach używa się krzyżyka. Opisane wcześniej zasady dotyczące nadawania nazw również mają tu zastosowanie. Poniższy przykład ustawia kolor tła oraz szerokość i wysokość banera:

```
#banner {
    background: #CC0000;
    height: 300px;
    width: 720px;
}
```

Identyfikatory w CSS stosuje się podobnie jak klasy, tylko używa się innego atrybutu — `id`. Przykładowo aby zastosować powyższy styl do elementu `<div>`, należałoby napisać taki kod HTML:

```
<div id="banner">
```

Aby zaznaczyć, że ostatni akapit strony stanowi informację o prawach autorskich, można utworzyć styl identyfikatora o nazwie `#copyright` i dodać go do tego akapitu:

```
<p id="copyright">
```

UWAGA Tak samo jak w przypadku klas, symbolu # używa się tylko przy nazwie stylu w arkuszu stylów. W wartości atrybutu znacznika HTML znak ten opuszczamy, na przykład `<div id="banner">`.

PORADNIK INŻYNIERA

Ukryte moce selektorów identyfikatora

Selektory identyfikatora mają kilka właściwości, których brak klasom. Nie mają one nic wspólnego z CSS, więc może nie będą nam nigdy potrzebne, ale skoro pytasz:

- W języku JavaScript identyfikatory są używane do lokalizacji sekcji strony i manipulacji nimi. Na przykład programiści często stosują identyfikatory w elementach formularzy, takich jak pola tekstowe, w których odwiedzający podaje swoje imię. Dzięki identyfikatorowi za pomocą JavaScriptu można uzyskać dostęp do takiego pola i w „magiczny" sposób sprawdzić, czy nie jest ono puste w momencie wysyłania formularza do serwera.
- Identyfikatory pozwalają także na tworzenie łączy do określonych części strony, co ułatwia nawigację po długich dokumentach. W przypadku glosariusza posortowanych alfabetycznie terminów za pomocą identyfikatorów można utworzyć odnośniki do konkretnych liter alfabetu. Kiedy użytkownik kliknie literę „R", zostanie natychmiast przeniesiony do miejsca, w którym znajdują się hasła zaczynające się na tę literę. Nie trzeba w tym przypadku pisać żadnych reguł CSS — wszystko odbywa się w HTML-u. Najpierw należy dodać identyfikator do miejsca na stronie, do którego ma prowadzić łącze. Na przykład w słowniku można zdefiniować elementy `<h2>` zawierające litery alfabetu, po których znajdują się definicje. Później do każdego z tych elementów `<h2>` należy dodać identyfikator: `<h2 id="R">R</h2>`. Tworząc odnośnik w HTML-u, na końcu adresu należy dodać krzyżyk i identyfikator, na przykład `index.html#R`. Łącze takie prowadzi bezpośrednio do elementu o identyfikatorze `#R` na stronie *index.html*. W tym zastosowaniu identyfikator jest podobny do połączeń nazwanych — `<a name="R">R</a>` — w HTML-u.

Nadawanie stylów grupom znaczników

Czasami potrzebny jest sposób na szybkie zastosowanie jednego stylu formatowania do kilku różnych elementów. Przykładowo można chcieć, aby wszystkie nagłówki na stronie miały taki sam kolor i krój pisma. Tworzenie oddzielnego stylu dla każdego z nich — `h1`, `h2`, `h3`, `h4` itd. — to zdecydowanie zbyt dużo pracy. Ponadto jeśli później zechcemy zmienić kolor tych nagłówków, trzeba będzie uaktualnić aż sześć różnych stylów. Lepszym rozwiązaniem będzie więc użycie selektora *grupowego*. Selektor grupowy pozwala na stosowanie stylów do wielu selektorów jednocześnie.

Grupowanie selektorów

Aby zgrupować selektory, wystarczy napisać ich listę rozdzieloną przecinkami. W związku z tym, żeby nadać ten sam kolor wszystkim nagłówkom, można utworzyć następującą regułę:

```
h1, h2, h3, h4, h5, h6 {color: #F1CD33;}
```

Powyższy przykład składa się z samych selektorów znacznika, ale można w ten sposób wykorzystać każdy prawidłowy selektor (lub kombinację typów selektorów). Przykładowo poniższy selektor nadaje ten sam kolor znacznikom `<h1>` i `<p>` należącym do klasy `.copyright` oraz temu, który ma identyfikator `#banner`.

```
h1, p, .copyright, #banner {color: #F1CD33;}
```

UWAGA Jeśli chcesz, aby kilka znaczników miało takie same *niektóre* style (ale nie wszystkie), to możesz utworzyć selektor grupowy ze wspólnymi właściwościami oraz indywidualne reguły z indywidualnymi właściwościami dla każdego znacznika osobno. Innymi słowy, do jednego znacznika może odnosić się kilka stylów. Możliwość stosowania wielu stylów do jednego elementu jest bardzo przydatną cechą CSS. Szczegóły na ten temat można znaleźć w rozdziale 5.

Selektor uniwersalny

Selektor grupowy można traktować jako skrócony zapis nadawania tego samego stylu kilku różnym elementom. W CSS dostępny jest także selektor wszystkich grup o nazwie selektor *uniwersalny*. Symbol gwiazdki (*) jest uniwersalnym selektorem pasującym do *wszystkich* znaczników.

Załóżmy, że chcemy na przykład, aby wszystkie znaczniki na stronie były pisane drukiem pogrubionym. Selektor grupowy w takiej sytuacji wyglądałby tak:

```
a, p, img, h1, h2, h3, h4, h5 itd. {font-weight: bold;}
```

Gwiazdka jest jednak o wiele krótszym sposobem na poinformowanie CSS, aby zastosował styl do wszystkich znaczników na stronie:

```
* {font-weight: bold;}
```

Selektora uniwersalnego można także użyć jako części selektora potomka, dzięki czemu można zastosować styl do wszystkich znaczników będących potomkami jednego określonego elementu strony. Na przykład selektor `.banner *` wybiera wszystkie znaczniki znajdujące się w elemencie należącym do klasy `banner` (selektory potomka są objaśnione w kolejnym podrozdziale).

Jako że selektor uniwersalny nie określa żadnego konkretnego typu znacznika, trudno przewidzieć, jaki wywoła efekt na wszystkich stronach witryny złożonych z wielu różnych znaczników HTML. Do formatowania wielu różnych elementów strony guru od projektowania stron posługują się *dziedziczeniem* — techniką CSS opisaną w następnym rozdziale.

Niektórzy projektanci stron internetowych wykorzystują selektor uniwersalny do usuwania *wszystkich* odstępów znajdujących się wokół elementów blokowych. Jak piszę w rozdziale 7., za pomocą własności `margin` można zdefiniować pustą przestrzeń wokół elementu, a za pomocą własności `padding` można ustawić odstęp między obramowaniem elementu i jego treścią. Przeglądarki automatycznie dodają różne ilości pustego miejsca wokół i wewnątrz różnych elementów, więc jednym ze sposobów na zapewnienie sobie czystej karty jest zastosowanie następującego stylu:

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

Stylizowanie znaczników zagnieżdżonych

Wybór dotyczący tego, czy używać na stronie selektorów znaczników, czy klas, wiąże się z pewnymi kompromisami. Selektory znaczników są szybkie i łatwe w użyciu, ale powodują, że dany znacznik zawsze wygląda tak samo, co nie jest problemem, jeśli chcemy, aby wszystkie nagłówki `<h2>` na stronie wyglądały identycznie. Selektory identyfikatorów i klas dają możliwość indywidualnego stylizowania elementów na stronie, w sposób niezależny od innych, ale tworzenie nowego stylu tylko po to, aby zmienić wygląd tekstu w nagłówku, wymaga bardzo dużo pracy i kodu HTML. To, co rozwiązałoby nasze problemy, to połączenie prostoty selektorów znacznika z precyzją klas i identyfikatorów. W CSS coś takiego nawet jest i nazywa się **selektorem potomka**.

Selektorów potomka używa się do formatowania znaczników „na pęczki" (dokładnie tak samo, jak w przypadku selektorów znacznika), ale tylko wtedy, gdy znajdują się one w określonej części strony. To tak, jakbyśmy mówili: „Hej, wy, wszystkie znaczniki `<a>` w pasku nawigacyjnym! Słuchajcie, mam dla was trochę formatowania. Wszystkie pozostałe znaczniki `<a>` mogą iść dalej. Nic tu po was".

Selektor potomka pozwala formatować znaczniki w oparciu o ich powiązania z innymi znacznikami. Aby zrozumieć jego działanie, trzeba nieco bardziej zagłębić się w sam HTML. Ponadto koncepcje leżące u podłoża selektorów potomka pomogą nam zrozumieć kilka innych selektorów, o których będziemy mówić nieco później.

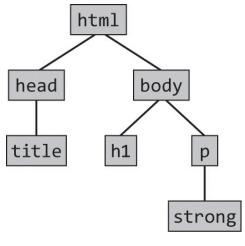
UWAGA Przy pierwszym zetknięciu selektory potomka mogą się wydawać skomplikowane i trudne do zrozumienia, ale należą one do najważniejszych technik CSS umożliwiających stosowanie stylów do elementów. Warto poświęcić trochę czasu, aby je opanować.

Drzewo rodzinne HTML

Kod HTML tworzący każdą stronę internetową można porównać do drzewa rodzinnego, na którym różne znaczniki HTML reprezentują różnych członków rodziny. Pierwszy znacznik użyty na stronie — `<html>` — jest dziadkiem wszystkich pozostałych znaczników. Zawiera on zarówno znacznik `<head>`, jak i `<body>`, dzięki czemu jest ich **przodkiem**. Również znacznik znajdujący się w innym znaczniku jest jego **potomkiem**. W poniższym przykładzie znacznik `<title>` jest potomkiem znacznika `<head>`:

```
<html>
  <head>
    <title>Prosty dokument</title>
  </head>
  <body>
    <h1>Nagłówek</h1>
    <p>Akapit <strong>ważnego</strong> tekstu.</p>
  </body>
</html>
```

Powyższy kod można przedstawić w postaci diagramu, jak na rysunku 3.4, uwidaczniając powiązania pomiędzy znacznikami na stronie. Pierwszy jest znacznik `<html>`. Dzieli się on na dwie sekcje reprezentowane przez znaczniki `<head>` i `<body>`. Zawierają one inne znaczniki, które z kolei mogą zawierać następne. Wiedząc, które znaczniki znajdują się w których, można każdą stronę przedstawić w postaci diagramu.



RYSUNEK 3.4.
Kod HTML składa się ze znaczników, w których zagnieżdżone są inne znaczniki zawierające kolejne znaczniki itd. Powiązania pomiędzy nimi — czyli to, w jaki sposób są one w sobie zagnieżdżone — tworzą coś w rodzaju drzewa rodzinnego

Drzewa pomagają zaobserwować, w jaki sposób CSS postrzega powiązania jednych elementów na stronie z innymi. Wiele selektorów opisywanych w tym rozdziale, włącznie z selektorem potomka, wykorzystuje te powiązania. Najważniejsze związki to:

- **Przodek** — jak pisałem na początku tego rozdziału, znacznik HTML zawierający inny znacznik jest jego przodkiem. Na rysunku 3.4 znacznik `<html>` jest przodkiem wszystkich pozostałych znaczników, natomiast znacznik `<body>` jest przodkiem wszystkich znaczników znajdujących się *w* nim — `<h1>`, `<p>` i `<strong>`.
- **Potomek** — znacznik znajdujący się w innym znaczniku lub innych znacznikach jest potomkiem. Na rysunku 3.4 znacznik `<body>` jest potomkiem znacznika `<html>`, podczas gdy `<title>` jest potomkiem znaczników `<head>` i `<html>`.

- **Rodzic** — rodzic jest *najbliższym* przodkiem znacznika. Na rysunku 3.4 rodzicem jest pierwszy znacznik bezpośrednio połączony z innym znacznikiem i znajdujący się nad nim. Tak więc znacznik `<html>` jest rodzicem tylko znaczników `<head>` i `<body>`. Znacznik `<p>` jest rodzicem znacznika `<strong>`.
- **Dziecko** — znacznik, który jest bezpośrednio w innym znaczniku. Na rysunku 3.4 zarówno znacznik `<h1>`, jak i `<p>` są dziećmi znacznika `<body>`, ale `<strong>` już nie, ponieważ jest on bezpośrednio wewnątrz znacznika `<p>`.
- **Brat** — znaczniki będące dziećmi jednego elementu są nazywane *braćmi*. Na diagramie HTML znaczniki braci znajdują się obok siebie i połączone są z tym samym rodzicem. Na rysunku 3.4 znaczniki `<head>` i `<body>` są braćmi, podobnie jak `<h1>` i `<p>`.

Na szczęście na tym kończą się w CSS odniesienia rodzinne. Nie musimy się już martwić o ciocie, wujków lub kuzynów (aczkolwiek plotka głosi, że w CSS10 *będą* nawet teściowe).

Tworzenie selektorów potomka

Selektory potomka pozwalają wykorzystać drzewo HTML, różnie formatując znaczniki w zależności od tego, czy znajdują się one w określonych innych znacznikach lub stylach. Przypuśćmy na przykład, że mamy znacznik `<h1>` i chcemy w nim wyróżnić jedno słowo za pomocą znacznika `<strong>`. Problem polega na tym, że większość przeglądarek wyświetla zarówno zawartość znacznika `<h1>`, jak i `<strong>` drukiem pogrubionym, przez co odwiedzający stronę nie zobaczą żadnej różnicy pomiędzy słowem wyróżnionym a resztą nagłówka. Zmiana koloru tekstu w znaczniku `<strong>` i tym samym wyróżnienie go za pomocą selektora znacznika nie jest dobrym rozwiązaniem, ponieważ spowoduje zmianę koloru *wszystkich* znaczników `<strong>` na stronie bez względu na to, czy nam się to podoba, czy nie. Selektor potomka pozwala zrobić to, co rzeczywiście chcemy, czyli zmienić kolor znacznika `<strong>` *tylko wtedy*, gdy znajduje się on w znaczniku `<h1>`.

Rozwiązanie dylematu ze znacznikami `<h1>` i `<strong>` wygląda tak:

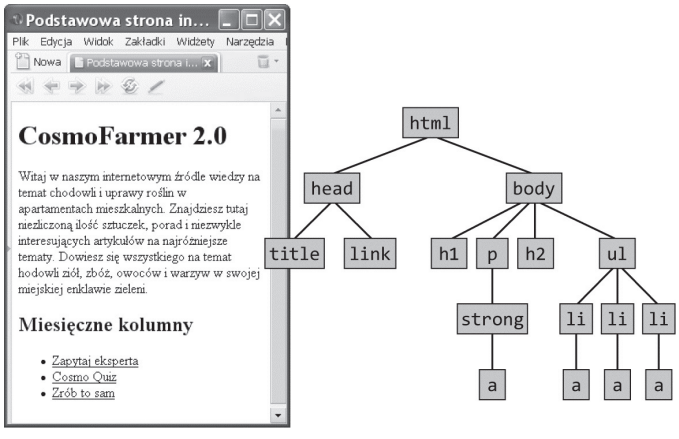
```
h1 strong {color: red;}
```

Zawartość *wszystkich* znaczników `<strong>` znajdujących się w znaczniku `<h1>` będzie miała kolor czerwony, a pozostałe znaczniki `<strong>` na stronie pozostaną nietknięte. Ten sam efekt można uzyskać przy użyciu klasy — na przykład `.strongHeader` — ale wówczas trzeba by było w dokumencie HTML dodać atrybut `class="strongHeader"` do elementów `<strong>` znajdujących się w nagłówkach. Dzięki selektorowi potomka nie trzeba nic zmieniać w kodzie HTML — wszystko robi się w arkuszu stylów!

Selektory potomka stylizują elementy, które są zagnieżdżone w innych elementach, posługując się dokładnie taką samą strukturą przodków i potomków jak w drzewie HTML. Selektor potomka tworzy się poprzez łączne użycie selektorów zgodnie z częścią drzewa, którą chcemy formatować. Przodek najwyższego rzędu (najstarszy) powinien znajdować się po lewej stronie, a selektor znacznika, który ma zostać sformatowany na samym końcu, po prawej stronie. Zwróćmy uwagę na przykład na trzy odnośniki (znacznik `<a>`) znajdujące się w liście

wypunktowanej i jeden odnośnik w akapicie na rysunku 3.5. Aby odnośniki w liście sformatować inaczej niż pozostałe na stronie, można utworzyć taki selektor potomka:

```
li a {font-family: Arial;}
```



RYSUNEK 3.5.
Ten prosty diagram (po prawej) reprezentuje strukturę strony internetowej widocznej po lewej stronie. Każdy znacznik na stronie jest potomkiem znacznika <html>, ponieważ w nim zawarta jest cała treść strony. Jeden znacznik może być potomkiem wielu innych znaczników. Na przykład pierwszy znacznik <a> widoczny na tym diagramie jest potomkiem znaczników , <p>, <body> i <html>

Reguła ta mówi: „Sformatuj wszystkie odnośniki (`a`) znajdujące się wewnątrz elementu listy (`li`), stosując krój pisma Arial". Selektor potomka może składać się z więcej niż dwóch elementów. Wszystkie poniższe przykłady są poprawnymi selektorami znaczników `<a>` znajdujących się w elementach listy na rysunku 3.5:

```
ul li a
body li a
html li a
html body ul li a
```

Powyższe cztery selektory — z których każdy robi to samo — pokazują, że nie trzeba wykorzystywać wszystkich przodków znacznika, który ma być sformatowany. Na przykład w drugim przykładzie (`body li a`) element `ul` nie jest potrzebny. Selektor ten zadziała, jeśli znacznik `<a>` będzie potomkiem (niekoniecznie bezpośrednim) znacznika `<li>` (który jest także potomkiem znacznika `<body>`). Selektor ten równie dobrze może odnosić się do znacznika `<a>` znajdującego się wewnątrz znacznika `<em>`, który jest w znaczniku `<strong>` będącym potomkiem znacznika `<li>` itd.

Ogólnie rzecz biorąc, selektory potomków powinny być jak najkrótsze. Jako że wszystkie elementy znajdują się w elementach `<html>` i `<body>`, nie ma potrzeby dodawać tych składników do selektora.

UWAGA Liczba indywidualnych selektorów w selektorze potomka ma znaczenie dla interakcji danego selektora z innymi kolidującymi z nim selektorami. Jest to związane z tzw. *precyzją* selektorów, o której piszę w rozdziale 5.

Nie ma także ograniczenia tylko do selektorów znaczników. Można tworzyć skomplikowane selektory potomka przy użyciu różnych typów selektorów.

Przypuśćmy na przykład, że chcemy, aby we wstępie (który oznaczony został klasą `intro`) odnośniki miały kolor żółty. Do tego celu służy poniższy selektor:

```
.intro a {color: yellow;}
```

Szybkie tłumaczenie: „Zastosuj ten styl do wszystkich odnośników (`a`), które są potomkami innego znacznika należącego do klasy `intro`".

Tworzenie modułów

Selektory potomka są często używane do formatowania **modułów** kodu, czyli zbiorów elementów HTML pełniących określoną rolę na stronie. Powiedzmy na przykład, że mamy na stronie element `<div>`, w którym prezentowane są najświeższe informacje dotyczące działalności firmy. Kod ten mógłby wyglądać tak:

```
<div>
    <h2>Nasza firma jest wspaniała!</h2>
    <p>Więcej informacji o tym, dlaczego nasza firma jest taka
    ↪znakomita</p>
    <h2>Kolejna wiadomość</h2>
    <p>Informacje o nowej wiadomości…</p>
    <h2>…itd…</h2>
    <p>…itd… </p>
</div>
```

Jeśli powyższemu elementowi `<div>` przypiszemy klasę — `<div class="news">` — to będziemy mogli pisać selektory potomka odnoszące się tylko do elementów wewnątrz tej sekcji wiadomości, na przykład:

```
.news h2 { color: red; }
.news p { color: blue; }
```

Teraz elementy `<h2>` znajdujące się w sekcji wiadomości będą czerwone, a akapity — niebieskie. Można nawet — i często się to robi — tworzyć selektory potomka składające się z kilku selektorów klasy. Wyobraź sobie na przykład, że tworzysz stronę internetową zawierającą wykaz adresów członków jakiejś organizacji. Każdy kontakt możesz umieścić w osobnym elemencie `<div>`, a następnie odpowiednio sformatować znajdujące się w nim inne elementy:

```
<div class="contact">
    <p class="name">Jan Kowalski</p>
    <p class="phone">555-555-1234</p>
    <p class="address">Klonowa 1234</p>
</div>
```

Następnie za pomocą selektorów potomka możesz sformatować tylko wybrane elementy kontaktu:

```
.contact .name { font-weight: bold; }
.contact .phone { color: blue ;}
.contact .address { color: red; }
```

Pseudoklasy i pseudoelementy

Czasami trzeba dobrać się do takich części strony, które nie są oznaczone żadnymi znacznikami, ale mimo wszystko łatwo je zlokalizować, jak na przykład pierwszy wiersz akapitu albo odnośnik w chwili, gdy umieszczono nad nim kursor. Do zabawy z tymi elementami w CSS służą *pseudoklasy* i *pseudoelementy*.

UWAGA W niektórych arkuszach stylów można znaleźć kod podobny do poniższego:

```
p.intro
```

Wygląda on podobnie do selektora potomka, ponieważ zawiera nazwę elementu HTML i klasę — ale nim nie jest. Między nazwą `p` i klasą `.intro` nie ma odstępu, co oznacza, że ten styl dotyczy tylko elementów `<p>` przypisanych do klasy `intro` (`<p class="intro">`). Dodanie spacji spowoduje zmianę działania tego selektora:

```
p .intro { color: yellow; }
```

Ta pozornie drobna zmiana powoduje, że styl zostanie zastosowany do wszystkich elementów należących do klasy `intro` i znajdujących się w elemencie `<p>`. Innymi słowy, ten selektor nie wybiera akapitu, tylko element znajdujący się w akapicie. Ogólnie lepiej jest opuszczać nazwę elementu HTML (tzn. pisać `.intro` zamiast `p.intro`), ponieważ dzięki temu style są bardziej elastyczne.

Style odnośników

Istnieją cztery pseudoklasy, które pozwalają formatować odnośniki będące w jednym z czterech stanów wywoływanych przez odwiedzającego. Sprawdzają one, kiedy odnośnik jest w jednym z czterech poniższych stanów:

- **`a:link`** — wybiera wszystkie jeszcze nieodwiedzone odnośniki, jeśli nie znajduje się nad nimi kursor i nie są właśnie klikane. Ten styl odpowiada za zwykły, nieodwiedzony odnośnik.
- **`a:visited`** — odnośnik, który był już wcześniej kliknięty, a informacja ta znajduje się w historii przeglądarki. Odnośnikowi tego typu można nadać inny styl niż zwykłym odnośnikom, mówiąc w ten sposób odwiedzającemu: „Hej, tutaj już byłeś” (w rozdziale 9., w ramce „Ograniczanie wolności” znajduje się opis ograniczeń zastosowania tego selektora).
- **`a:hover`** — pozwala na zmianę wyglądu odnośnika w momencie najechania na niego myszą. Efekty najechania myszą na odnośnik nie są tworzone tylko dla zabawy — mogą one także dostarczać przydatnych informacji dla przycisków na pasku nawigacyjnym.

  Pseudoklasy `:hover` można używać także z innymi elementami niż odnośniki. Za jej pomocą można przykładowo wyróżnić tekst w znaczniku `<p>` albo `<div>`, kiedy odwiedzający najedzie na niego myszą. W takim przypadku zamiast selektora `a:hover` (który służy do formatowania łączy) możesz użyć selektora `p:hover`. Jeśli chcesz, aby efekt dotyczył tylko elementów należących do wybranej klasy, możesz tę klasę dodać przed pseudoklasą, na przykład: `.highlight:hover`.
- **`a:active`** — pozwala określić wygląd odnośnika w momencie kliknięcia go. Innymi słowy, odpowiada za krótką chwilę od momentu wciśnięcia przycisku myszy do jego zwolnienia.

Rozdział 9. zawiera opis sposobu projektowania wyglądu odnośników przy użyciu omówionych pseudoklas w taki sposób, aby ułatwić użytkownikom nawigację po stronie.

UWAGA Bez selektorów opisanych w następnych podrozdziałach można spokojnie sobie poradzić. Wielu projektantów nigdy ich nie używa. Opisane dotychczas selektory — znacznika, klasy, identyfikatora, potomka, grupowy itd. — pozwalają na tworzenie pięknych, funkcjonalnych i łatwych w utrzymaniu stron. Jeśli jesteś gotowy rozpocząć prawdziwą zabawę, czyli projektowanie stron, to możesz pominąć kurs z dalszej części rozdziału. Można zawsze do niego wrócić w jakiś zimny deszczowy dzień spędzany przed kominkiem.

Stylizowanie fragmentów akapitu

Cechy typograficzne, dzięki którym książki i czasopisma wyglądają ładnie i elegancko, nie istniały w początkowej fazie rozwoju sieci (bo kiedy niby naukowcy martwili się o to, aby coś wyglądało ładnie?). W CSS dostępne są dwa pseudoelementy `:first-letter` i `:first-line`, które nadają stronom internetowym finezji znanej od wieków w druku.

Pseudoelement `:first-letter` umożliwia kontrolę wyglądu pierwszej litery akapitu, która wyróżnia się na tle reszty tekstu większym lub grubszym pismem, jak na przykład na początku rozdziału.

Nadanie odmiennego koloru pierwszej linii tekstu w akapicie (`:first-line`) przyciąga wzrok i daje wrażenie świeżości (jeśli Cię to zaintrygowało, to rozdział 6. jest w całości poświęcony formatowaniu tekstu i zawiera także opis tych dwóch selektorów).

UWAGA W CSS3 zmieniono składnię pseudoelementów. W CSS 2.1 poprzedzało się je jednym dwukropkiem:

```
:first-letter
```

Natomiast w CSS3 należy używać dwóch dwukropków, aby odróżnić pseudoelementy od pseudoklas, na przykład `:hover`. W związku z tym teraz zamiast `:first-letter` i `:first-line` należy pisać `::first-letter` i `::first-line`. Na szczęście przeglądarki nadal obsługują także starą wersję zapisu pseudoelementów. To bardzo dobra wiadomość, szczególnie jeśli weźmie się pod uwagę fakt, że Internet Explorer 8 nie rozpoznaje składni z dwoma dwukropkami. Dlatego też na razie lepiej pozostać przy składni z jednym dwukropkiem.

Więcej pseudoklas i pseudoelementów

W specyfikacji CSS można znaleźć wiele innych ciekawych i bardzo przydatnych pseudoklas i pseudoelementów. Selektory te są bardzo dobrze obsługiwane przez wszystkie nowsze przeglądarki.

PSEUDOKLASA :FOCUS

Pseudoklasa `:focus` działa podobnie jak `:hover`. Podczas gdy `:hover` ma zastosowanie, gdy użytkownik najedzie myszą na odnośnik, `:focus` reaguje na zdarzenie, podczas którego użytkownik zrobi coś, co wskazuje na jego zainteresowanie danym elementem, z reguły jest to kliknięcie lub zaznaczenie za pomocą klawisza *Tab*. To kliknięcie jest jedyną wskazówką dla projektanta dotyczącą tego, na czym dany użytkownik skupił swoją uwagę.

Selektor `:focus` przydaje się głównie do prezentowania różnych informacji użytkownikowi, na przykład zmiana koloru tła może wskazywać, gdzie należy wpro-

wadzić tekst (selektor ten jest często stosowany do pojedynczych pól tekstowych, pól haseł i ramek z tekstem `<textarea>`). Poniższy styl zmienia na jasnożółty kolor każdego pola tekstowego, w którym kliknie użytkownik lub które zostanie zaznaczone klawiszem *Tab*:

```
input:focus {background-color: #FFFFCC;}
```

Selektor `:focus` ma zastosowanie tylko wtedy, gdy na danym elemencie jest skoncentrowana uwaga odwiedzającego. Jeśli kliknie on w innym polu tekstowym lub w ogóle gdzieś indziej na stronie, to punkt koncentracji zostanie przeniesiony z tego pola wraz z właściwościami CSS.

WSKAZÓWKA Dobrym źródłem informacji na temat poziomu obsługi selektorów przez różne przeglądarki jest strona *www.caniuse.com*.

PSEUDOELEMENT :BEFORE

Pseudoelement `:before` potrafi zrobić coś, czego nie umie żaden inny selektor. Pozwala dodać treść przed dowolnym wybranym elementem. Załóżmy na przykład, że przed niektórymi akapitami chcemy umieścić ramkę „GORĄCA WSKAZÓWKA" podobną do „PRZYSPIESZAMY" i „PORADNIK INŻYNIERA" z tej książki. Zamiast wpisywać ten tekst na stronach HTML, można wyręczyć się pseudoelementem `:before`. Pozwala to nie tylko oszczędzić na pisaniu kodu, lecz także, jeśli zdecydujemy się zmienić nazwę z „GORĄCA WSKAZÓWKA" na „TO TRZEBA WIEDZIEĆ", sprawić, że zmiany na wszystkich stronach zostaną wprowadzone dzięki tylko jednej małej zmianie w arkuszu stylów. Wadą tej metody jest to, że wstawiony w ten sposób tekst nie będzie widoczny dla przeglądarek nieobsługujących CSS lub selektora `:before`.

Najpierw trzeba utworzyć klasę (powiedzmy `.tip`) i zastosować ją do akapitów, przed którymi chcemy umieścić nasz tekst: `<p class="tip">`. Następnie dodajemy tekst do arkusza stylów:

```
.tip:before {content: "GORĄCA WSKAZÓWKA"}
```

Za każdym razem, gdy przeglądarka napotka klasę `.tip` w jakimkolwiek elemencie, posłusznie wstawi przed nim tekst `GORĄCA WSKAZÓWKA`.

Tekst dodawany za pomocą pseudoelementu `:before` technicznie nazywa się **treścią generowaną dynamicznie**, ponieważ jest tworzony w locie przez przeglądarkę. W kodzie HTML strony tekst ten nie istnieje. Bez względu na to, czy sobie to uświadamiamy, czy nie, przeglądarki cały czas generują własną treść, jak punkty w listach wypunktowanych czy cyfry w listach numerowanych. Za pomocą pseudoelementu `:before` można nawet określić sposób prezentacji punktów i cyfr w listach.

Internet Explorer od wersji 8 i inne najważniejsze przeglądarki dobrze obsługują pseudoelement `:before` (jak również opisany poniżej selektor `:after`).

PSEUDOELEMENT :AFTER

Pseudoelement `:after` dodaje treść generowaną dynamicznie, ale, w przeciwieństwie do pseudoelementu `:before`, za wybranym elementem, a nie przed. Za jego pomocą po cytowanych słowach można wstawić cudzysłów zamykający (`"`).

UWAGA Zarówno `:before`, jak i `:after` to pseudoelementy, podobnie jak `:first-line` i `:first-letter`. Jak napisałem wcześniej, w CSS3 zmieniono składnię pseudoelementów, tak że zamiast jednego dwukropka używa się dwóch, a więc zamiast `:before` i `:after` należy pisać `::before` i `::after`. Na szczęście przeglądarki nadal obsługują także starą wersję składni, dzięki czemu można używać selektorów `:before` i `:after`, które są obsługiwane przez Internet Explorera 8.

PSEUDOELEMENT ::SELECTION

Ten selektor CSS3 odnosi się do elementów, które zostały zaznaczone na stronie przez użytkownika. Gdy na przykład przeglądający stronę gdzieś kliknie i przeciągnie myszą, w przeglądarce zostanie zaznaczony fragment tekstu, który można następnie skopiować. Normalnie tło takiego zaznaczenia jest niebieskie. W Internet Explorerze zmienia się też kolor tekstu na biały. Za pomocą omawianego selektora można zmienić te ustawienia. Gdybyś na przykład chciał, aby zaznaczony tekst był biały i miał fioletowe tło, mógłbyś napisać następującą regułę CSS:

```
::selection {
    color: #FFFFFF;
    background-color: #993366;
}
```

Przy użyciu tego selektora można definiować tylko własności `color` i `background-color`, a więc nie możesz pójść na całość i zmienić rozmiaru czcionki, kroju pisma, marginesów i dokonać innych zmian, które doprowadzałyby Twoich czytelników do szału (dziękujemy CSS za to, że chroni nas przed nami samymi).

UWAGA Pseudoelement `::selection` występuje tylko w wersji z dwoma dwukropkami, a więc zapis `:selection` jest niepoprawny i nie zadziała.

Selektor `::selection` nie jest obsługiwany przez przeglądarki Internet Explorer 8, Firefox i iOS Safari. W Firefoksie można sobie poradzić, stosując wersję selektora **z przedrostkiem**:

```
::-moz-selection {
    color: #FFFFFF;
    background-color: #993366;
}
```

Aby formatowanie zaznaczonego tekstu działało w Firefoksie i innych przeglądarkach, w arkuszu stylów należy umieścić oba rodzaje reguł — jedna pod drugą (więcej informacji o przedrostkach przeglądarek znajduje się w rozdziale 7.).

Jeśli chcesz zaszaleć, możesz zmienić tło zaznaczenia tylko w wybranych elementach. Na przykład poniższy selektor odnoszący się do elementów `<p>` zmienia kolor tekstu akapitów na czerwony, a tła — na różowy.

```
p::selection {
    color: red;
    background-color: pink;
}
```

WSKAZÓWKA Nauka pisania selektorów czasami przypomina naukę pisania hieroglifów. Na stronie *http://gallery.theopalgroup.com/selectoracle* można przetłumaczyć dowolny selektor na wersję w naturalnym języku (niestety, nie ma języka polskiego). Wystarczy wpisać dowolnie długi selektor, a program zwróci opis jego działania w prostym angielskim.

Selektory atrybutu

CSS umożliwia formatowanie znaczników na podstawie posiadanych przez nich atrybutów. Przypuśćmy na przykład, że chcemy, aby obrazy na naszej stronie były otoczone ramką — ale tylko te najważniejsze. Nie chcemy, aby zabiegowi temu były poddane logo, przyciski i inne dodatki, które także korzystają ze znacznika `<img>`. Na szczęście do każdego ważnego obrazu dodaliśmy opis za pomocą atrybutu `title`. Oznacza to, że za pomocą **selektora atrybutu** można zidentyfikować wszystkie te obrazy.

Dzięki selektorom atrybutów można odnosić się do znaczników o określonych właściwościach. Na przykład poniższy selektor wybiera wszystkie znaczniki `<img>` z atrybutem `title`:

```
img[title]
```

Pierwsza część selektora to nazwa znacznika (`img`), a nazwa atrybutu znajduje się w nawiasach kwadratowych (`[title]`).

Atrybuty selektorów nie są ograniczone do nazw znaczników. Można stosować także klasy. Na przykład selektor `.photo[title]` dopasowuje się do wszystkich elementów klasy `.photo`, które mają atrybut HTML `title`.

Aby uzyskać jeszcze większą precyzję, można wybrać elementy, które nie tylko mają określony atrybut, ale także określony atrybut o określonej wartości. Przykładowo jeśli chcemy wyróżnić odnośniki do konkretnego adresu URL, to możemy utworzyć przyciągający wzrok selektor atrybutu, jak ten poniżej:

```
a[href="http://www.cafesoylentgreen.com"]{
    color:red;
    font-weight:bold;
}
```

Dodawanie wartości do selektora atrybutu jest bardzo przydatne podczas pracy z formularzami. Wiele elementów formularzy ma takie same znaczniki, chociaż wyglądają i zachowują się całkiem różnie. Pola wyboru i tekstowe, a także przyciski wysyłania formularza, wykorzystują ten sam znacznik `<input>`. Formę i przeznaczenie takiego pola określa wartość atrybutu `type`. Przykładowo kod `<input type="text">` tworzy pole tekstowe, a `<input type="checkbox">` pole wyboru.

Aby wybrać tylko pola tekstowe formularza, można na przykład użyć takiego selektora:

```
input[type="text"]
```

Selektor atrybutu jest bardzo wszechstronny. Za jego pomocą można wybierać nie tylko elementy mające określony atrybut o określonej wartości (np. wszystkie pola wejściowe typu `checkbox`), ale również elementy, których określony atrybut ma wartość *zaczynającą się* i *kończącą się* określonym łańcuchem znaków lub go *zawierającą*. Mimo że może się to wydawać przesadą, jest bardzo przydatne.

Wyobraź sobie na przykład, że chcesz utworzyć styl do wyróżniania łączy zewnętrznych (tzn. prowadzących do stron spoza Twojej witryny), aby czytelnik wiedział, że kliknięcie ich spowoduje opuszczenie jego serwisu. Jeśli do połączeń między podstronami witryny nie używasz adresów bezwzględnych, możesz przyjąć, że łącza zewnętrzne zaczynają się od ciągu znaków `http://`.

Wówczas selektor odnoszący się do tych odnośników wyglądałby tak:

```
a[href^="http://"]
```

Ciąg ^= oznacza początek wartości atrybutu, a więc selektor odnosi się do wszystkich odnośników, których adres zaczyna się od znaków `http://`. Przypisane mu deklaracje zostałyby zastosowane do adresu *http://www.google.com*, *http://www.sawmac.com* itd. Innymi słowy, selektor ten wybiera łącza zewnętrzne.

UWAGA Przedstawiony selektor nie zadziała dla łączy do bezpiecznych stron, czyli takich, których adres zaczyna się od znaków *https://*. Aby sformatować także te łącza, można zastosować grupowanie selektorów:

```
a[href^="http://"], a[href^=" https://"]
```

Czasami trzeba wybrać elementy, których atrybut ma określony łańcuch znaków *na końcu* wartości. Wyobraź sobie na przykład, że przy łączach prowadzących do dokumentów PDF chcesz umieścić specjalną ikonę. Ponieważ dokumenty PDF mają rozszerzenie *.pdf*, wiadomo, że łącza do nich kończą się właśnie tym łańcuchem znaków, na przykład: `<a href="download_instructions.pdf">`. Selektor dotyczący tylko tego typu odnośników wyglądałby tak:

```
a[href$=".pdf"]
```

A cały styl mógłby wyglądać następująco:

```
a[href$=".pdf"] {
    background: url(doc_icon.png) no-repeat;
    padding-left: 15px;
};
```

Nie przejmuj się własnościami użytymi w tej regule — własność `padding` poznasz w rozdziale 7., a o ustawianiu obrazów w tle elementów jest mowa w rozdziale 8. W tej chwili ważne jest to, że ciąg `$=` oznacza koniec wartości atrybutu. W podobny sposób można utworzyć selektory odnoszące się do dokumentów Word (`[a href$=".doc"]`), filmów (`a [href$=".mp4"]`) itd.

Można też wybierać elementy na podstawie środka wartości ich atrybutów. Wyobraź sobie, że publikujesz w witrynie zdjęcia pracowników firmy i chcesz je jakoś wyróżnić, na przykład zdefiniować im grube zielone obramowanie i szare tło. Możesz w tym celu każdemu zdjęciu przypisać specjalną klasę, na przykład `.headshot`, ale to wymaga sporo ręcznej pracy. Jeśli nazwy zdjęć są tworzone według jakiegoś wzoru, z problemem możesz uporać się znacznie sprawniej.

Powiedzmy, że nazwa każdego zdjęcia zawiera słowo *headshot*, na przykład *mcfarland_headshot.jpg*, *mccord_headshot.jpg*, *headshot_albert.jpg* itd. Każda z tych nazw zawiera słowo *headshot*, co oznacza, że zawiera je również atrybut `src` każdego elementu `<img>`. Selektor wybierający tylko te elementy jest przedstawiony poniżej:

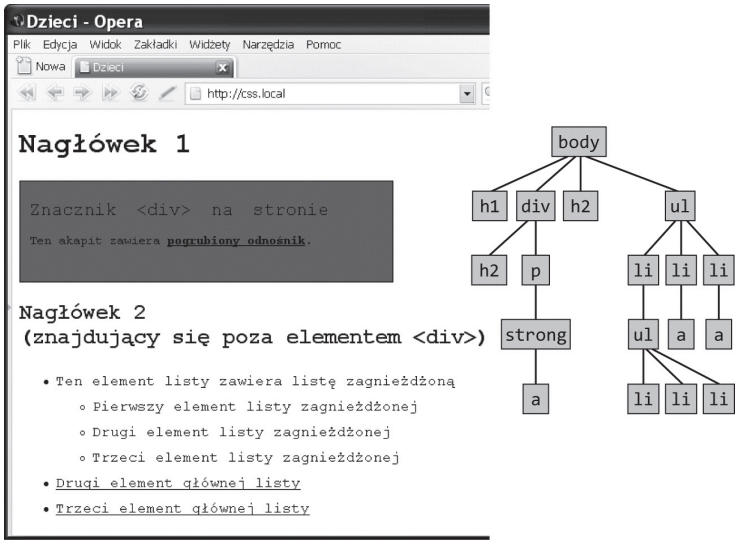
```
img[src*="headshot"]
```

Kod ten można przeczytać tak: „Wybierz wszystkie obrazy, których atrybut `src` zawiera słowo `headshot` w jakimkolwiek miejscu". Jest to prosty i elegancki sposób na sformatowanie wszystkich obrazów.

Selektor dziecka

Podobnie jak opisany wcześniej selektor potomka, **selektor dziecka** w CSS pozwala na formatowanie dzieci znaczników. Selektor ten korzysta z dodatkowego symbolu nawiasu ostrego (>) w celu oznaczenia związku pomiędzy dwoma elementami. Przykładowo selektor `body > h1` wybiera wszystkie znaczniki `<h1>` będące dziećmi znacznika `<body>`.

W przeciwieństwie do selektora potomka, który odnosi się do *wszystkich* potomków danego znacznika (dzieci, wnuków itd.), selektor dziecka pozwala określić, o które dziecko którego rodzica chodzi. Na przykład na rysunku 3.6 są dwa znaczniki `<h2>`. Użycie zwykłego selektora potomka `body h2` spowoduje wybranie obu tych znaczników. Mimo że każdy z nich znajduje się wewnątrz znacznika `<body>`, to jego dzieckiem jest tylko ten drugi. Pierwszy znacznik `<h2>` znajduje się bezpośrednio w znaczniku `<div>`, a więc jest on jego rodzicem. Jako że znaczniki te mają różnych rodziców, dostęp do każdego z nich można uzyskać oddzielnie. Aby wybrać tylko ten drugi znacznik `<h2>`, selektor powinien wyglądać tak: `body > h2`. Dostęp zaś do pierwszego z nich można uzyskać za pomocą selektora `div > h2`.



RYSUNEK 3.6. *Diagram ten (po prawej) pokazuje powiązania pomiędzy znacznikami HTML (po lewej)*

W CSS3 dostępnych jest kilka bardzo konkretnych pseudoklas do wybierania elementów dziecka. Dzięki nim można pisać selektory wybierające elementy z różnych drzew HTML.

PSEUDOKLASA :FIRST-CHILD

Wróćmy na chwilę do analogii do drzewa rodzinnego HTML i przypomnijmy sobie, co to jest znacznik dziecka: jest to każdy znacznik znajdujący się w innym znaczniku (na rysunku 3.6 znaczniki `<h1>`, `<p>`, `<h2>` i `<ul>` są dziećmi znacznika

<body>). Pseudoklasa :first-child pozwala formatować wszystkie elementy będące *pierwszym* dzieckiem danego elementu.

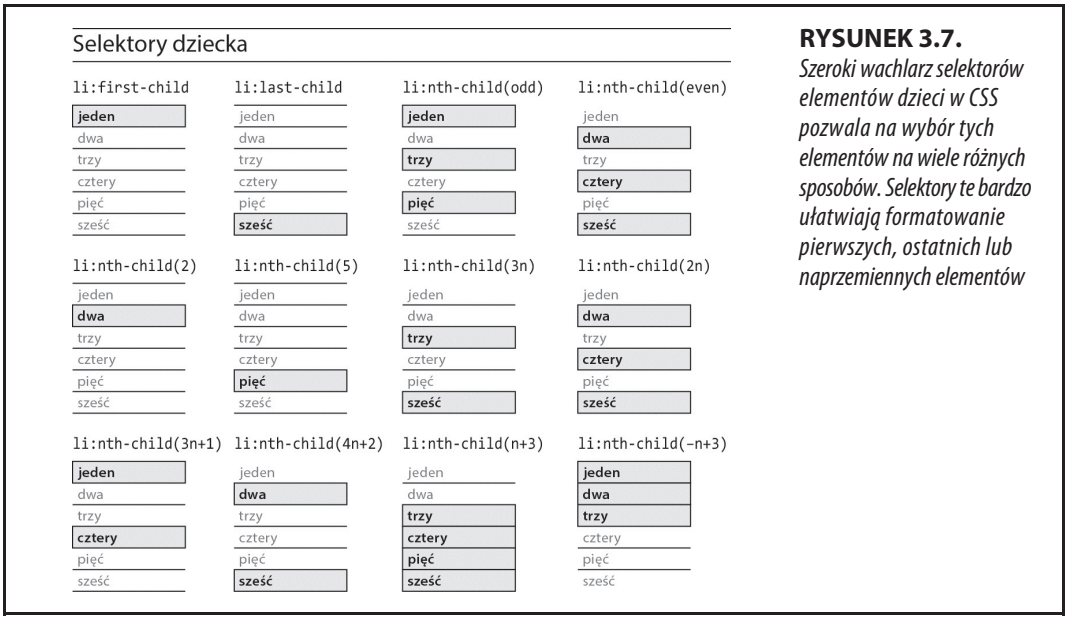
Aby wybrać pierwszy element <h1> na rysunku 3.6, należy napisać następujący selektor:

```
h1:first-child
```

Ten selektor wybiera wszystkie elementy <h1> będące pierwszym dzieckiem innego elementu. W dokumencie przedstawionym na rysunku 3.6 efekt jego zastosowania jest oczywisty, ponieważ znajduje się w nim tylko jeden taki element, który na dodatek jest pierwszym dzieckiem elementu <body>. Jednak czasami działanie pseudoklasy :first-child może być niejasne. Jeśli na przykład zmienisz element <h2> znajdujący się w elemencie na rysunku 3.6 na <h1>, to selektor :first-child wybierze zarówno element <h1> znajdujący się bezpośrednio w elemencie <body>, jak i element <h1> w kontenerze (ponieważ <h1> jest pierwszym dzieckiem elementu).

PSEUDOELEMENT :LAST-CHILD

Ten pseudoelement działa podobnie do pseudoelementu :first-child, tylko dotyczy ostatniego elementu dziecka. Aby na przykład sformatować ostatni element listy, należy napisać ul :last-child (rysunek 3.7).



RYSUNEK 3.7. *Szeroki wachlarz selektorów elementów dzieci w CSS pozwala na wybór tych elementów na wiele różnych sposobów. Selektory te bardzo ułatwiają formatowanie pierwszych, ostatnich lub naprzemiennych elementów*

PSEUDOELEMENT :ONLY-CHILD

Istnieje też selektor elementu będącego jedynym dzieckiem innego elementu. Powiedzmy na przykład, że mamy w arkuszu stylów taką regułę:

```
p:only-child {
  color: red;
}
```

Zawarta w tej regule deklaracja zmienia kolor tekstu na czerwony, ale wyłącznie wtedy, gdy w danym elemencie znajduje się tylko jeden akapit. Powiedzmy, że mamy na stronie element `<div>` z trzema akapitami. W tym przypadku powyższy selektor nie zadziała, ponieważ blok `<div>` zawiera trzy elementy `<p>`. Gdybyśmy jednak dwa z nich usunęli, to ten jedyny pozostały stałby się czerwony.

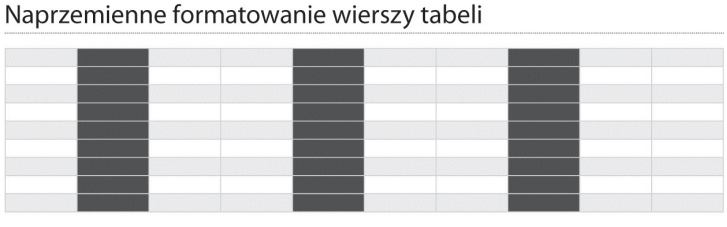
Aby było jeszcze trudniej, selektor ten działa tylko wtedy, gdy określony element jest w ogóle jedynym dzieckiem innego elementu. Innymi słowy, nie wystarczy, że element jest jedynym przedstawicielem danego typu. Jeśli ma jakiegokolwiek brata, selektor nie zadziała. Gdybyśmy na przykład obok elementu `<p>` dodali do bloku `<div>` listę `<ul>`, to akapit nie byłby już jedynym dzieckiem. Element `<ul>` też nim jest, więc selektor `p:only-child` w tym przypadku straciłby moc!

PSEUDOKLASA :NTH-CHILD

Jest to dość skomplikowany, ale niezwykle przydatny selektor. Za jego pomocą można na przykład sformatować co drugi wiersz tabeli, co trzeci element listy albo zdefiniować jeszcze jakiś inny sposób naprzemiennego formatowania elementów (rysunek 3.7). Elementy do sformatowania określa się za pomocą specjalnych wartości. Najłatwiej jest użyć słów kluczowych `even` i `odd`, które odnoszą się — odpowiednio — do parzystych i nieparzystych elementów dzieci. Aby na przykład parzystym wierszom tabeli nadać jeden kolor tła, a nieparzystym inny, można napisać poniższe dwie reguły CSS:

```
tr:nth-child(odd) { background-color: #D9F0FF; }
tr:nth-child(even) { background-color: #FFFFFF; }
```

To bardzo łatwy sposób na naprzemienne pokolorowanie wierszy tabeli (rysunek 3.8). Jednak to nie wszystkie możliwości selektora `:nth-child()`.



RYSUNEK 3.8. *Naprzemienne formatowanie kolumn tabeli można zrealizować w łatwy sposób przy użyciu selektora dziecka. Można formatować kolumny naprzemiennie, a nawet co trzecią, zaczynając od drugiej (jak na tym rysunku). To się nazywa precyzyjna kontrola*

Za jego pomocą można też wybrać dziecko o określonym numerze. Gdybyśmy na przykład chcieli sformatować piąty element listy, to moglibyśmy wpisać w nawiasie selektora `:nth-child` liczbę `5`:

```
li:nth-child(5)
```

Ten selektor wybiera tylko pojedynczy element. Gdybyśmy natomiast chcieli sformatować np. co trzeci element listy, w wartości wpisalibyśmy liczbę (w tym przypadku `3`) i literę `n`:

```
li:nth-child(3n)
```

Litera `n` jest mnożnikiem, więc `3n` znaczy co trzeci element dziecko, zaczynając od trzeciego (rysunek 3.7).

A co zrobić, jeśli chcielibyśmy wybrać co trzeci element dziecko, zaczynając od drugiego? Powiedzmy, że chcemy wyróżnić co trzecią komórkę w wierszu tabeli (element <td>), zaczynając od drugiej (rysunek 3.8). Poniżej znajduje się odpowiednia reguła:

```
td:nth-child(3n+2) { background-color:#900; }
```

Liczba znajdująca się przed literą `n` (w przykładzie `3`) reprezentuje mnożnik, tzn. `3n` oznacza co trzeci element, a `4n` — co czwarty. Natomiast po znaku `+` (`+2` w przykładzie) wpisuje się numer startowy odliczania, a więc `+2` oznacza, że chcemy zacząć liczenie od drugiego elementu, a `+5` — od piątego. W związku z tym selektor `:nth-child(5n+4)` wybiera co piąty element dziecko, zaczynając od czwartego.

Wartość `n` może być nawet ujemna i wówczas selektor wybiera elementy *od końca*. Na przykład ostatnia lista na rysunku 3.7 jest formatowana za pomocą poniższego selektora:

```
li:nth-child(-n+3)
```

Selektor ten można przeczytać tak: „Przejdź do trzeciego elementu na liście, następnie wybieraj każdy element listy znajdujący się przed nim". Jak widać, selektor `nth-child` jest dość skomplikowany, ale za jego pomocą można tworzyć praktycznie nieskończoną liczbę odwołań do różnych elementów.

Selektory dziecka z uwzględnieniem typu elementu

W CSS3 dodano selektor, który działa podobnie do opisanych wcześniej selektorów dziecka, ale dotyczy elementów konkretnego *typu*. Wyobraź sobie, że chcesz sformatować pierwszy akapit w pasku bocznym, ale na niektórych stronach pierwszym dzieckiem tego paska jest element <h2>, a na innych <p>. Nie można użyć selektora `:first-child`, aby wybrać element akapitu, ponieważ w niektórych przypadkach akapit ten jest *drugim* dzieckiem (znajduje się za elementem <h2>). Jest on (element <p>) jednak pierwszym akapitem w tym pasku bocznym, dzięki czemu można go wybrać za pomocą selektora **`:first-of-type`**.

UWAGA Selektory `:last-child`, `:first-of-type` oraz `:nth-child()` są obsługiwane przez wszystkie nowoczesne przeglądarki, wliczając Internet Explorer od wersji 9. Nie działają natomiast w Internet Explorerze 8.

PSEUDOKLASA :FIRST-OF-TYPE

Pseudoklasa `:first-of-type` działa podobnie jak `:first-child`, ale odnosi się do elementu dziecka określonego typu. Wyobraź sobie, że masz na stronie pasek boczny przypisany do klasy `sidebar`. Aby sformatować pierwszy akapit w tym pasku, należy użyć poniższego selektora:

```
.sidebar p:first-of-type
```

Nazwę interesującego nas elementu podajemy przed pseudoelementem: `p:first-of-type` — tu odnosimy się do elementu <p>.

PSEUDOKLASA :LAST-OF-TYPE

Ta pseudoklasa działa podobnie jak `:last-child`, ale odnosi się do ostatniego elementu określonego typu. Przykładowo jeśli chcesz sformatować ostatni akapit w pasku bocznym `<div>`, ale nie masz pewności, czy za tym akapitem nie będzie żadnych innych elementów (np. listy wypunktowanej, nagłówka albo obrazu), możesz użyć poniższego selektora:

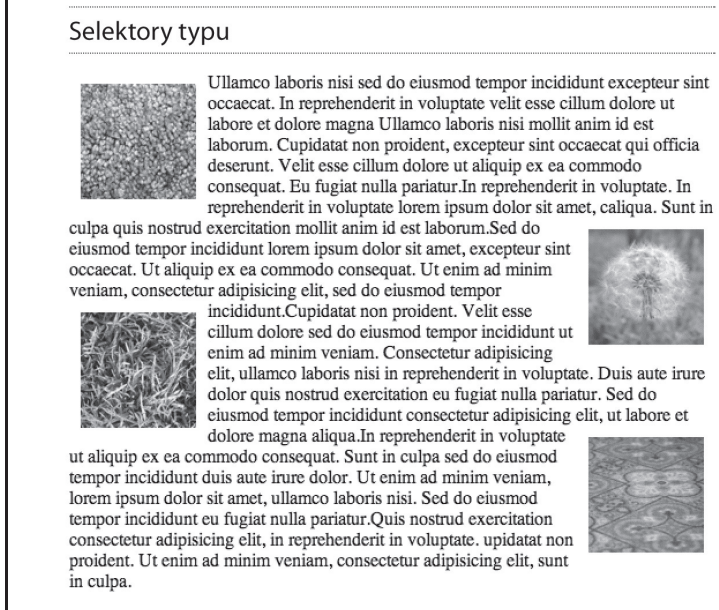
```
.sidebar p:last-of-type
```

UWAGA Pamiętaj, że te selektory typu dotyczą elementów będących dziećmi innych elementów, a więc `p:first-of-type` oznacza „pierwszy element dziecko będący akapitem".

PSEUDOKLASA :NTH-OF-TYPE

Ta pseudoklasa działa podobnie do `:nth-child()`, ale odnosi się do naprzemiennych elementów dzieci określonego typu. Może się przydać, gdy na stronie znajdują się duże akapity zawierające obrazy. Element `<img>` jest śródliniowy, a więc może być zagnieżdżany w akapicie w dowolnej liczbie. Teraz wyobraź sobie, że chcesz, aby zdjęcia w akapitach były naprzemiennie przesunięte do lewej lub prawej krawędzi, jak widać na rysunku 3.9. Efekt ten można osiągnąć za pomocą poniższych selektorów:

```
img:nth-of-type(odd) { float: left; }
img:nth-of-type(even) { float: right; }
```



RYSUNEK 3.9. *Za pomocą selektora :nth-of-type() można łatwo wybrać co drugi obraz w kontenerze i naprzemiennie przenieść obrazy pod lewą i prawą krawędź tego kontenera*

Jak widać, dostępne są takie same słowa kluczowe (`odd` i `even`) oraz obowiązuje taka sama składnia (`2n+1`) jak w przypadku selektora `:nth-child()`.

Tak naprawdę za pomocą selektora `:nth-of-type()` można również naprzemiennie sformatować wiersze tabeli:

```
tr:nth-of-type(odd) { background-color: #D9FOFF; }
tr:nth-of-type(even) { background-color: #FFFFFF; }
```

Przy użyciu selektorów CSS do każdego elementu zawsze można dostać się na więcej niż jeden sposób — zazwyczaj jest pięć i więcej sposobów!

CZĘSTO ZADAWANE PYTANIE

Tworzenie pięknych list

Kiedy mogę potrzebować selektora dziecka? Po lekturze tego rozdziału do tej pory znam już wystarczająco dużo selektorów, aby uzyskać dostęp do każdego elementu na stronie. Po co więc uczyć się jeszcze jednego selektora?

Jest jeden problem w projektowaniu stron, który najlepiej rozwiązuje się za pomocą selektorów dziecka. Występuje on na większej liczbie stron, niż sobie wyobrażasz. Mając nieuporządkowaną listę z zagnieżdżoną jedną lub większą liczbą innych list nieuporządkowanych (jak na rysunku 3.6), można użyć selektorów dziecka do wizualnego zorganizowania tych kategorii i podkategorii informacji. Elementy listy pierwszego poziomu można sformatować w jeden sposób, a elementy drugiego poziomu w inny. Treść zaprezentowana w ten sposób wygląda schludnie, profesjonalnie i jest czytelna (a użytkownicy pokochają nas za to).

Najpierw utwórzmy klasę dla listy najwyższego poziomu i nazwijmy ją `.mainList`. Do tego najwyższego poziomu można zastosować bezszeryfowy krój pisma oraz nieco powiększyć tekst względem reszty strony i pogrubić go lub nadać mu inny kolor. Następne kategorie mogą być pisane mniejszą szeryfową czcionką, jak na przykład Times, co zwiększy czytelność. W przypadku dużej ilości tekstu nadanie nieco innego stylu każdej podkategorii pomoże użytkownikom zorientować się w nim.

Zastosuj klasę `.mainList` do pierwszego znacznika `<ul>` (`<ul class="mainList">`). Następnie za pomocą selektora dziecka (`ul.mainList > li`) wybierz pierwszy zestaw elementów listy i nadaj im żądany styl. Formatowanie to będzie miało zastosowanie tylko do znaczników `<li>` będących dziećmi znacznika `<ul>` należącego do klasy `.mainList`. Znaczniki `<li>` będące dziećmi pozostałych zagnieżdżonych znaczników `<ul>` pozostaną nienaruszone, dzięki czemu można nadać im właściwe formatowanie, niezależne od tego wcześniej, przy użyciu odpowiednich selektorów dziecka. Na przykład aby nadać styl znacznikom `<li>` będącym dziećmi pierwszej zagnieżdżonej listy, należy użyć selektora `ul.main ↪List > li > ul > li` (dla porównania selektor potomka, jak `ul li`, wybiera wszystkie elementy wszystkich list nieuporządkowanych na stronie — zarówno tych zagnieżdżonych, jak i niezagnieżdżonych).

Będziesz musiał pamiętać o pojęciu, które poznasz w następnym rozdziale, czyli dziedziczeniu. Mówiąc krótko, niektóre własności CSS po zastosowaniu do jednego elementu HTML są dziedziczone przez elementy znajdujące się w tym elemencie. Nawet jeśli użyjesz selektora dziecka, aby sformatować wybrany element, użyte własności mogą przejść na inne elementy, które się w nim znajdują. Jednym ze sposobów na uniknięcie tego jest użycie selektora `:not()`.

Selektor brata

Powiązania typu rodzic-dziecko nie są jedynym rodzajem związków w drzewie HTML. Czasami zachodzi potrzeba uzyskania dostępu do znacznika nie na podstawie jego związku z elementem nadrzędnym, a na podstawie innych znaczników, które mają tego samego rodzica co on. Znacznik znajdujący się bezpośrednio obok innego znacznika w HTML-u nazywa się jego **bratem** (ang. *adjacent sibling*). Na rysunku 3.6 znacznik `<div>` jest bratem znacznika `<h1>`, znacznik `<p>` jest bratem `<h2>` itd.

Dzięki selektorowi brata można na przykład pierwszemu akapitowi znajdującemu się po każdym nagłówku nadać inne formatowanie niż następnym akapitom. Przypuśćmy, że chcemy usunąć margines znajdujący się nad tym znacznikiem `<p>`, aby umieścić go bezpośrednio pod nagłówkiem bez żadnego odstępu. Można też nadać mu inny kolor lub rozmiar czcionki, tworząc coś w rodzaju wstępu.

Selektor brata wykorzystuje do łączenia elementów symbol plusa (+). Aby więc wybrać wszystkie akapity występujące po znaczniku <h2>, należy użyć selektora `h2 + p` (spacje nie są obowiązkowe, więc równie dobrze można napisać `h2+p`). Ostatni element w tym selektorze to ten, któremu zostanie nadany styl — ale tylko wtedy, gdy jego bratem jest znacznik <h2>.

Jest jeszcze jeden selektor brata — o nazwie **ogólny kombinacyjny selektor brata** (spróbuj to powiedzieć szybko kilka razy z rzędu). Ma on postać tyldy (~) i oznacza: „Wybierz wszystkich braci tego typu". Na przykład podczas gdy selektor `h2+p` wybiera jeden element <p> znajdujący się bezpośrednio za elementem <h2>, selektor `h2 ~ p` wybierze *wszystkie* elementy `p` znajdujące się za elementem <h2> na tym samym poziomie drzewa dokumentu. Możliwe, że ten selektor nigdy Ci się nie przyda, ale kaskadowe arkusze stylów są znane ze swoich dużych możliwości.

Selektor :target()

Selektor `:target()` jest bardzo zabawny. Za jego pomocą można uzyskać bardzo ciekawe efekty, np. zastosować styl do jednego elementu *po tym*, jak inny element zostanie kliknięty. Tego rodzaju interaktywne funkcje do tej pory zawsze kojarzyły się z JavaScriptem. Działanie selektora `:target()` opiera się na atrybutach identyfikatora, których opis znajduje się w ramce „Ukryte moce selektorów identyfikatora". Identyfikator służy jako połączenie z określonym miejscem na stronie.

Mamy na przykład stronę internetową o nazwie *index.html*. Na stronie tej znajduje się element o identyfikatorze `signupForm`. Jeśli na tej lub innej stronie znajduje się odnośnik wskazujący `index.html#signupForm`, to jego kliknięcie spowoduje przejście przez przeglądarkę do tego elementu . Jeżeli element ten znajduje się na dole bardzo długiej strony, to przeglądarka przewinie ją. Czasami odnośniki takie nazywa się łączami **wewnątrz strony** (ang. *in page link*) i często wykorzystuje się je do budowy glosariuszy itp. stron, na których użytkownik może kliknąć literę, aby przejść do wybranego miejsca słownika.

Ale funkcji tej można też używać do innych celów. Jeśli adres URL w pasku adresu przeglądarki zawiera znak # i identyfikator, to element opatrzony tym identyfikatorem staje się celem, dzięki czemu można zastosować dowolny styl do elementu tylko wtedy, gdy jego identyfikator znajduje się w adresie URL.

Przyjrzymy się działaniu selektora `:target` na konkretnym przykładzie. Wyobraź sobie, że ten kod znajduje się gdzieś w pobliżu początku strony internetowej:

```
<button>
  <a href="#signupForm">Zapisz się do newslettera</a>
</button>
<form id="signupForm">
  <label for="email">Adres e-mail</label>
  <input type="email" id="email">
  <input class="btn" type="submit" value="Zapisz">
</form>
```

Gdy użytkownik kliknie ten odnośnik — element <a> — celem stanie się formularz. Innymi słowy, możemy sprawić, że formularz w zwykłym stanie będzie formatowany przez jeden zestaw stylów, a w stanie po kliknięciu przez użytkownika

odnośnika — przez drugi zestaw stylów. Początkowo możemy na przykład ukryć formularz (więcej na temat ustawiania widoczności elementów piszę w rozdziale 14.):

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

Reguła ta powoduje ukrycie formularza tak, że bezpośrednio po wczytaniu strony nie będzie on widoczny. Ale gdy użytkownik kliknie odnośnik *Zapisz się do newslettera*, formularz staje się celem i możemy go pokazać za pomocą poniższej reguły:

```
#signupForm:target {
  display: block;
}
```

Innymi słowy, jeżeli adres URL w pasku adresu przeglądarki zawiera tylko nazwę pliku — np. *index.html* — to przeglądarka zastosuje dla formularza pierwszą regułę stylistyczną, która spowoduje, że będzie on niewidoczny. Ale gdy końcówka adresu będzie miała postać *index.html#signupForm*, to zastosowanie będzie miał styl `:target`, co spowoduje pokazanie formularza na stronie. Fajna rzecz.

UWAGA Bardzo ciekawe przykłady użycia selektora `:target` można obejrzeć na stronie *http://benschwarz.github.io/gallery-css/*.

Selektor :not()

Selektor `:not()`, czasami nazywany też **pseudoklasą negacji**, umożliwia określanie, których elementów nie chcemy wybrać do formatowania. Przykładowo do akapitów możemy zastosować jakąś klasę, na przykład `<p class="classy">`, a następnie napisać regułę CSS do jej sformatowania:

```
.classy { color: red; }
```

Ale co, jeśli chcielibyśmy wybrać wszystkie akapity *oprócz* tych, które są przypisane do klasy `classy`? Do tego właśnie przyda nam się selektor `:not()`. W jego nawiasie należy wpisać selektor dotyczący elementu, którego *nie chcemy* wybrać, na przykład:

```
p:not(.classy) { color: blue; }
```

Powyższa reguła zmieni kolor tekstu na niebieski we wszystkich akapitach oprócz należących do klasy `classy`.

Selektor `:not()` może być bardzo przydatny podczas używania opisanych wcześniej selektorów atrybutów. W sekcji „Selektory atrybutu" znajduje się przykład selektora atrybutu wybierającego wszystkie łącza zewnętrzne:

```
a[href^="http://"]
```

Jak wiesz, selektor ten nie wybiera jednak łączy prowadzących do zewnętrznych stron, a jedynie łącza zaczynające się od znaków `http://`. W wielu witrynach to

jedno i to samo, ponieważ do zewnętrznych stron tworzone są odnośniki z adresami bezwzględnymi, a do wewnętrznych połączeń — adresy względne. Czasami jednak adresy bezwzględne używane są także do połączeń wewnętrznych.

Na przykład niektóre systemy zarządzania treścią (np. WordPress) tworzą przy użyciu adresów bezwzględnych odnośniki do wpisów na blogu. W takich przypadkach, aby formatowane były tylko łącza prowadzące na zewnątrz, konieczne jest poprawienie selektora atrybutu poprzez użycie selektora `:not()`. Wyobraź sobie, że Twoja witryna ma domenę *mojastrona.pl*. Aby sformatować łącza prowadzące poza tę domenę, musisz wybrać odnośniki zawierające adresy bezwzględne, które *nie* wskazują na domenę *mojastrona.pl*:

```
a[href^="http://"]:not([href^="http://mojastrona.pl"])
```

Powyższy selektor można przeczytać tak: „Wybierz wszystkie łącza, których atrybut `href` ma wartość zaczynającą się od znaków `http://`, ale *pomiń* te, które zaczynają się od ciągu `http://mojastrona.pl`". Przypomnę, że w selektorze atrybutu zapis ^= oznacza początek wartości atrybutu. Omawiany selektor można też zapisać krócej:

```
a[href^="http://"]:not([href*="mojastrona.pl "])
```

W selektorze atrybutu zapis *= oznacza „zawiera", a więc wykluczyliśmy wszystkie adresy bezwzględne zawierające łańcuch `mojastrona.pl`. Zaliczają się do nich zarówno `http://www.mojastrona.pl`, jak i `http://mojastrona.pl`.

Istnieją pewne ograniczenia dotyczące zastosowania selektora `:not()`:

- Można z nim używać tylko **selektorów prostych**, czyli selektorów elementów (np. `html` albo `p`), selektora uniwersalnego (*), klas (np. `.footer`), identyfikatorów (np. `#banner`) oraz pseudoklas (`:hover`, `:checked`, `:first-child` itd.). Wszystkie poniższe selektory są poprawne:

  ```
  .footnote:not(div)
  img:not(.portrait)
  div:not(#banner)
  li:not(:first-child)
  ```

- Nie można z nim używać selektorów potomka (np. `div p a`), pseudoelementów (np. `::first-line`), selektorów grupowych ani kombinatorów (np. selektora brata `h2+p`).

- Nie można łączyć selektorów `:not()` w łańcuchy. Na przykład poniższy selektor jest niepoprawny:

  ```
  a[href^="http://"]:not([href*="google.com"]):not([href="yahoo.com])
  ```

  Innymi słowy, selektora `:not()` można użyć tylko raz w selektorze złożonym.

Kurs: selektory

Pozostała część tego rozdziału została poświęcona tworzeniu różnego rodzaju selektorów i sprawdzaniu ich wpływu na wygląd strony. Zaczniemy od najprostszych stylów, stopniowo przechodząc do coraz bardziej zaawansowanych.

Do rozpoczęcia pracy potrzebne będą pliki dostępne pod adresem *ftp://ftp.helion.pl/przyklady/cssnp4.zip*. Pobierz te pliki i, jako że są one spakowane w archiwum ZIP, rozpakuj je. Pliki dotyczące tego kursu znajdują się w folderze o nazwie *03*.

PYTANIE WZBURZONEGO CZYTELNIKA

Wszystko w jednym pliku

Hej, o co chodzi z tym wewnętrznym arkuszem stylów w tym kursie? W rozdziale 2. wymieniono masę powodów do używania zewnętrznych arkuszy stylów?

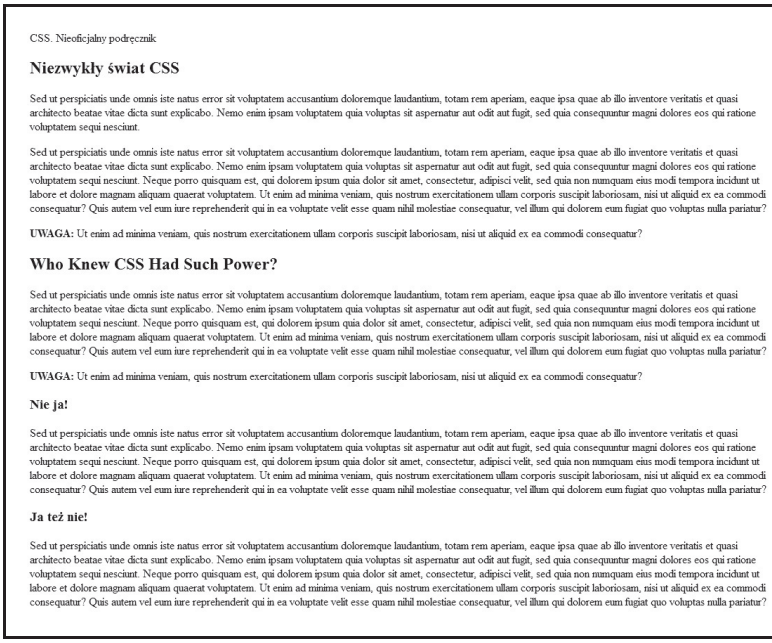
Myślisz, że jesteś strasznie mądry? Tak, zewnętrzne arkusze stylów zazwyczaj pozwalają tworzyć szybsze i bardziej wydajne strony, o czym pisałem już w rozdziale 2. Jednak wewnętrzne arkusze stylów ułatwiają życie, kiedy pracuje się nad pojedynczą stroną, jak w tym kursie. Dzięki temu zamiast bez przerwy przełączać się pomiędzy zewnętrznym arkuszem a plikiem HTML, można cały czas pracować w jednym wspólnym pliku. Ponadto można podglądać postęp bez ciągłego odświeżania pamięci podręcznej przeglądarki. Więcej na temat tych zawiłości napisałem w rozdziale 2., w ramce „Nie daj się złapać na pamięć podręczną".

Krótko mówiąc, do tworzenia stron internetowych powinno się używać zewnętrznych arkuszy stylów. Gdybyśmy chcieli użyć stylów z tego kursu w więcej niż jednym pliku HTML, to ułatwiłoby nam to pracę. Aby jednak ułatwić sobie naukę CSS, będziemy posługiwać się jednym plikiem HTML i osadzonym w nim arkuszem stylów.

Niemniej jednak to było bardzo dobre pytanie!

1. Otwórz w swoim ulubionym edytorze plik *03/selector_basics.html*.

Strona ta składa się z kilku podstawowych znaczników HTML. W tym kursie nieco ją ożywimy. Zaczniemy od dodania fontu Google, którego użyliśmy już w rozdziale 2.



CSS. Nieoficjalny podręcznik

Niezwykły świat CSS

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

UWAGA: Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur?

Who Knew CSS Had Such Power?

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

UWAGA: Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur?

Nie ja!

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

Ja też nie!

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

RYSUNEK 3.10. *Czysty HTML w przeglądarce wygląda nieprzyjemnie i monotonnie. Ale dzięki CSS brzydkie kaczątko (tutaj) można zamienić w łabędzia (rysunek 3.11) w trzydziestu jeden prostych krokach*

2. **W pustym wierszu za zamykającym znacznikiem `</title>` wpisz poniższy kod:**

```
<link href='https://fonts.googleapis.com/css?family=Bitter&subset=latin,
↪latin-ext' rel='stylesheet' type='text/css'>
```

Jak napisałem w rozdziale 2., element ten dołącza do strony zewnętrzny arkusz stylów z serwerów firmy Google. Pobiera ona font Bitter, którego dzięki temu możesz używać na swojej stronie. (Szerzej na temat dołączania do stron fontów sieciowych z serwerów Google piszę w rozdziale 6.). Następnie dodamy wewnętrzny arkusz stylów.

3. **Za dodanym w poprzednim punkcie elementem `<link>` naciśnij klawisz *Enter* i wpisz `<style>`.** Naciśnij *Enter* jeszcze dwa razy i wpisz `</style>`.

Są to znaczniki otwierający i zamykający arkusza stylów — dobrze jest od razu wpisywać znacznik zamykający razem z otwierającym, aby nie zapomnieć tego zrobić później. Znaczniki te informują przeglądarkę, że kod znajdujący się między nimi to kaskadowe arkusze stylów. Kod HTML powinien wyglądać teraz tak (dodany kod jest wyróżniony):

```
<title>Podstawy selektorów</title>
<link href='https://fonts.googleapis.com/css?family=Bitter&subset=latin,
↪latin-ext' rel='stylesheet' type='text/css'>
<style>

</style>
```

Selektory typu — jak selektor znacznika, który za chwilę utworzymy — są podstawowym rodzajem selektorów (ci, którzy ukończyli kurs z poprzedniego rozdziału, musieli utworzyć już kilka takich selektorów).

4. **Kliknij między znacznikami stylu i wpisz `body {`, naciśnij dwa razy klawisz *Enter* i wpisz `}`.**

Dobrze jest zawsze od razu zamykać reguły CSS, aby później o tym nie zapomnieć. Aby utworzyć selektor znacznika, wystarczy posłużyć się nazwą znacznika HTML, który ma być sformatowany. Ten styl odnosi się do całego dokumentu zawartego w elemencie `<body>`. Teraz możemy ustawić kolor tła i marginesy całej strony.

5. **Kliknij między klamrami stylu i wpisz poniższe deklaracje CSS w celu nadania stylów — koloru, rozmiaru i kroju pisma oraz wcięcia (jako dopełnienia):**

```
body {
  background-color: rgb(50,122,167);
  padding: 0 20px 20px 20px;
  margin: 0;
}
```

Za pomocą klawisza *Enter* umieść każdą właściwość w osobnym wierszu. Dobrze jest też wizualnie sformatować kod CSS, wcinając każdą właściwość przy użyciu tabulatora (niektórzy zamiast tabulatora używają dwóch spacji, ale to tylko kwestia gustu).

Użyte w tym przykładzie własności zmieniają kolor tła strony — `rgb()` to jeden ze sposobów na definiowanie wartości barwy czerwonej, zielonej

i niebieskiej. W tym przypadku został zdefiniowany ciemnoniebieski kolor. To sprawia, że czarny tekst jest słabo widoczny, więc musimy zmienić kolor tekstu akapitów.

UWAGA Te właściwości i ich wartości mogą wydawać się niezbyt znajome. Na razie po prostu je wpisz. W ten sposób dowiesz się wstępnie, jak się ustawia marginesy i dopełnienie. W rozdziale 6. poznasz natomiast szczegóły na ten temat.

6. Dodaj kolejny styl pod regułą dotyczącą elementu `<body>`.

```
p {
  color: rgba(255,255,255,.6);
  font-size: 1em;
  font-family: "Varela Round", Arial, Helvetica, sans-serif;
}
```

Deklaracje te określają formatowanie wszystkich akapitów (elementów `<p>`) — kolor, rozmiar pisma i krój pisma. Tym razem kolor zdefiniowałem za pomocą funkcji `rgba()`. Znajdujący się na końcu człon `a` służy do określania poziomu przezroczystości koloru. W tym przypadku ustawiliśmy biały kolor tekstu (255,255,255 w formacie RGB) o 60-procentowej „nieprzezroczystości" (wartość `.6`). Dzięki temu przez tekst przebija się nieco niebieskiego koloru tła, co sprawia, że tekst wygląda na błękitny.

Czas sprawdzić efekty naszej pracy.

7. Otwórz stronę w przeglądarce, aby zobaczyć wynik swojej pracy.

Jeśli nic nie jest poprzestawiane w ustawieniach, większość przeglądarek wyświetla czarny tekst w standardowym szeryfowym kroju pisma, jak Times. Jeśli style CSS działają jak trzeba, to na stronie powinno być widać siedem akapitów napisanych krojem Bitter o miłym dla oka jasnoniebieskim kolorze.

Tworzenie selektora grupowego

Czasami kilka elementów na stronie ma ten sam wygląd. Przykładowo dla zachowania jednolitego stylu można wszystkim nagłówkom nadać ten sam kolor. Zamiast tworzyć oddzielne style i powielać te same ustawienia właściwości dla każdego znacznika `<h1>`, `<h2>` itd., można je zebrać w jednym selektorze grupowym.

1. Wróć do swojego edytora i pliku *selector_basics.html*.

Pod wcześniej utworzonym stylem znacznika `<p>` dodamy nowy styl.

2. Kliknij za zamykającą klamrą selektora znacznika, naciśnij klawisz *Enter*, aby przejść do nowego wiersza, a następnie wpisz poniższy kod:

```
h1, h2, h3 {
}
```

Jak wyjaśniałem wcześniej, selektor grupowy to po prostu lista selektorów oddzielonych przecinkami. Ta reguła nada takie samo formatowanie, które dodamy za chwilę, wszystkim znacznikom `h1`, `h2` i `h3` na stronie.

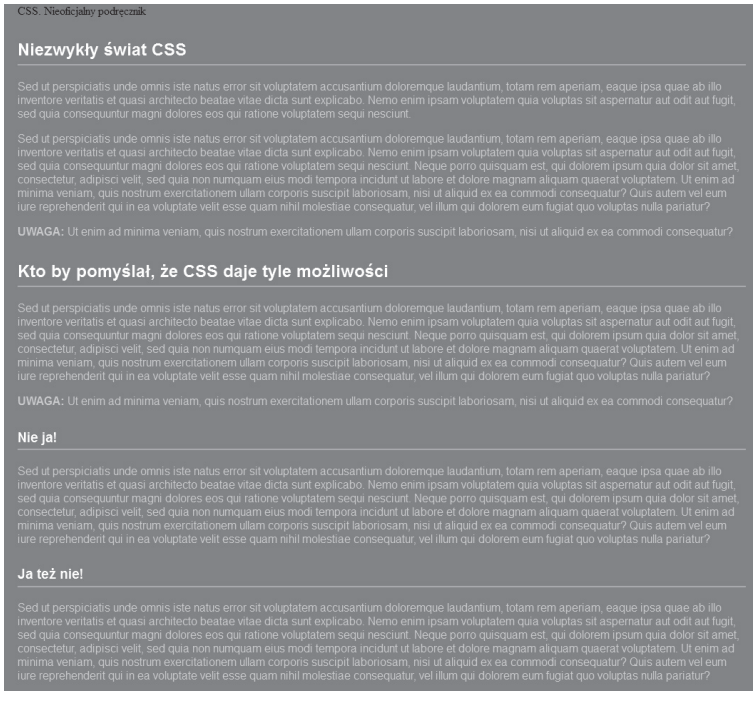
3. **Kliknij w pustym wierszu między klamrami i dodaj pięć poniższych deklaracji:**

```
color: rgb(255,255,255);
font-family: Arial, "Palatino Linotype", Times, serif;
border-bottom: 2px solid rgb(87,185,178);
padding-top: 10px;
padding-bottom: 5px;
```

Sporo tego kodu, ale tak naprawdę ustawiamy w nim tylko kolor i krój pisma nagłówków, definiujemy u góry linię obramowania mającą zwiększyć ich walory wizualne oraz ustawiamy górne i dolne dopełnienie mające odsunąć tekst od krawędzi elementu. Własność `padding` odsuwa treść od krawędzi elementu, nie wpływając w żaden sposób na jego krawędzie ani tło — po prostu dodajemy trochę pustego miejsca nad i pod nagłówkiem.

4. **Zapisz plik i obejrzyj wynik w przeglądarce.**

Nagłówek `<h1>` znajdujący się w pobliżu górnej krawędzi strony oraz nagłówki `<h2>` i `<h3>` będące tytułami sekcji mają taki sam kolor i krój pisma oraz zieloną linię na dole (rysunek 3.11). Nagłówek `<h1>` wydaje się trochę za mały, ale łatwo można go trochę powiększyć.



RYSUNEK 3.11. *Prosty selektor typu może całkowicie zmienić wygląd każdego wystąpienia danego elementu na stronie, co znacznie usprawnia formatowanie np. wszystkich akapitów na stronie. W tym przypadku użyliśmy nawet selektora grupowego, dzięki któremu sformatowaliśmy trzy różne rodzaje nagłówków naraz!*

5. **Wróć do edytora tekstu i pliku *selector_basics.html*. Dodaj jeszcze jedną regułę pod stylem z selektorem grupowym:**

```
h1 {
  font-size: 2em;
}
```

Zwiększamy rozmiar pisma nagłówków pierwszego stopnia. Domyślna wielkość `font-size` stosowana przez przeglądarki wynosi `1em`, więc zmieniliśmy ją na dwa razy większą. Zauważ też, że do elementu może się odnosić kilka stylów naraz — na przykład w tym przypadku reguły z selektorami `h1, h2, h2` i `h1`. Zarówno pierwsza, jak i druga z tych reguł mają zastosowanie do wszystkich elementów `<h1>` na stronie. Nazywa się to *kaskadą*, o której więcej dowiesz się w rozdziale 5.

Tworzenie i stosowanie selektora identyfikatora

Selektory identyfikatorów służą do formatowania pojedynczych elementów na stronie. Tworzy się specjalną regułę CSS i przypisuje się elementowi HTML specjalny atrybut, dzięki któremu deklaracje z tej reguły zostaną przypisane do wybranego elementu. Identyfikatorów często używa się do oznaczania elementów formularzy, tworzenia odnośników do wybranych części stron (zobacz ramkę „Ukryte moce selektorów identyfikatora" we wcześniejszej części rozdziału) oraz do pracy z elementami za pomocą skryptów JavaScript.

Choć wielu projektantów stron internetowych odradza stosowanie tego selektora (szczegóły podaję w rozdziale 5.), to i tak warto go znać.

W tym ćwiczeniu utworzymy styl określający wygląd tekstu znajdującego się na samej górze strony — napisu *CSS. Nieoficjalny podręcznik*. Ma to być logo strony, więc nadamy mu też identyfikator.

1. **Wróć do swojego edytora i pliku *selector_basics.html*.**

   Dodamy nowy styl pod utworzoną wcześniej regułą `h1`.

2. **Kliknij za zamykającą klamrą poprzedniego stylu i naciśnij klawisz *Enter*, aby przejść do nowego wiersza, a następnie wpisz `#logo {`.**

   Selektory identyfikatora zawsze zaczynają się od znaku krzyżyka (`#`). Nazwa tego stylu wskazuje, że dotyczy on logo strony.

3. **Jeszcze raz naciśnij klawisz *Enter* i wpisz poniższy kod:**

```
font-family: Baskerville, Palatino, sans-serif;
font-size: 2em;
color: rgba(255,255,255,.8);
font-style: italic;
text-align: center;
margin-bottom: 30px;
background-color: rgb(191,91,116);
border-radius: 0 0 10px 10px;
padding: 10px;
```

   Jest to dość długa lista deklaracji, ale za ich pomocą ustawiamy tylko pewne właściwości pisma, kolor tła oraz pustą przestrzeń wokół logo.

4. **Zakończ styl, wpisując zamykającą klamrę. Całość powinna wyglądać tak:**

```
#logo {
  font-family: Baskerville, Palatino, sans-serif;
  font-size: 2em;
  color: rgba(255,255,255,.8);
  font-style: italic;
  text-align: center;
```

```
  margin-bottom: 30px;
  background-color: rgb(191,91,116);
  border-radius: 0 0 10px 10px;
  padding: 10px;
}
```

Jeśli zapiszesz plik i podejrzysz go w przeglądarce, to nie zauważysz żadnych zmian. Jest to spowodowane tym, że reguła *nie* działa, dopóki nie zostanie do czegoś zastosowana. W związku z tym teraz dodamy wybranemu elementowi HTML atrybut identyfikatora o odpowiedniej wartości.

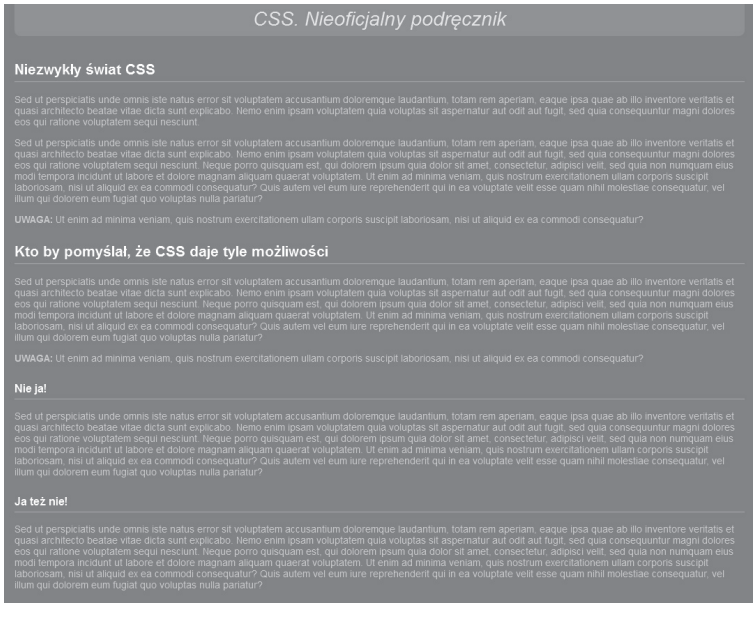
5. **Znajdź w kodzie strony otwierający znacznik `<div>` znajdujący się w pobliżu znacznika `<body>` — powinien zawierać tekst *CSS. Nieoficjalny podręcznik* — i dodaj do niego atrybut `id="logo"`:**

```
<div id="logo">
  CSS. Nieoficjalny podręcznik
</div>
```

Teraz element `<div>` będzie formatowany zgodnie z deklaracjami przypisanymi do selektora `#logo`. W CSS każdy efekt można osiągnąć na wiele sposobów. W tym przypadku ten sam efekt można uzyskać przy użyciu klasy zamiast identyfikatora, niemniej jednak użyjemy identyfikatora, ponieważ służy on do oznaczania pojedynczych elementów strony, takich jak np. logo.

6. **Zapisz stronę i obejrzyj wynik w przeglądarce.**

Teraz tekst „CSS. Nieoficjalny podręcznik" znajduje się na środku strony, ma jasny kolor oraz otacza go ramka przylegająca do górnej krawędzi strony (rysunek 3.12).



RYSUNEK 3.12. *Użycie selektora identyfikatora jest jednym ze sposobów formatowania pojedynczych elementów na stronie, takich jak logo na górze tej strony*

Tworzenie i stosowanie selektora klasy

Selektory typu są szybkie i efektywne, ale brakuje im wybiórczości, jeśli chodzi o nadawanie stylów na stronie. Co zrobić, jeśli chcemy, aby jeden znacznik <p> wyróżniał się spośród pozostałych znaczników <p> na stronie? Odpowiedzią jest selektor klasy.

1. **Przejdź z powrotem do swojego edytora i pliku *selector_basics.html*.**

   Dodamy nowy styl pod utworzonym przed chwilą selektorem grupowym.

2. **Kliknij za zamykającą klamrą reguły z selektorem `#logo`, naciśnij *Enter* i wpisz poniższy kod:**

   ```
   .note {
   }
   ```

   Nazwa tego stylu (`note`) wskazuje jego przeznaczenie — wyróżnić akapity zawierające dodatkowe informacje dla odwiedzających stronę. Po utworzeniu stylu klasy można go zastosować w dowolnym miejscu — w tym przypadku jest to trzeci akapit.

3. **Kliknij w pustym wierszu między klamrami i dodaj do stylu poniższe deklaracje:**

   ```
   color: black;
   border: 2px solid white;
   background-color: rgb(69,189,102);
   margin-top: 25px;
   margin-bottom: 35px;
   padding: 20px;
   ```

   Zwróć uwagę na wartość koloru w formacie `rgb()`. W CSS kolory można definiować na kilka sposobów, w tym także za pomocą słów kluczowych (`white`, `black` lub `orange`). Więcej na ten temat dowiesz się w rozdziale 6.

   Jeśli teraz podejrzysz stronę, to nie zobaczysz żadnych zmian. Selektory klas, podobnie jak selektory identyfikatorów, działają dopiero wtedy, gdy zdefiniuje się odpowiednie atrybuty w kodzie HTML.

4. **W kodzie HTML strony znajdują się dwa elementy <p>, których treść zaczyna się od słowa „Uwaga” otoczonego znacznikami .**

   Aby zastosować klasę do znacznika, wystarczy wstawić do niego atrybut `class`, którego wartością jest nazwa selektora klasy — w tym przypadku dodamy nazwę utworzonego przed chwilą stylu `.note`.

5. **Kliknij zaraz za `p` w znaczniku <p>, a następnie naciśnij jeden raz spację i wpisz `class="note"`. Kod HTML powinien teraz wyglądać tak (nowy kod jest wyróżniony):**

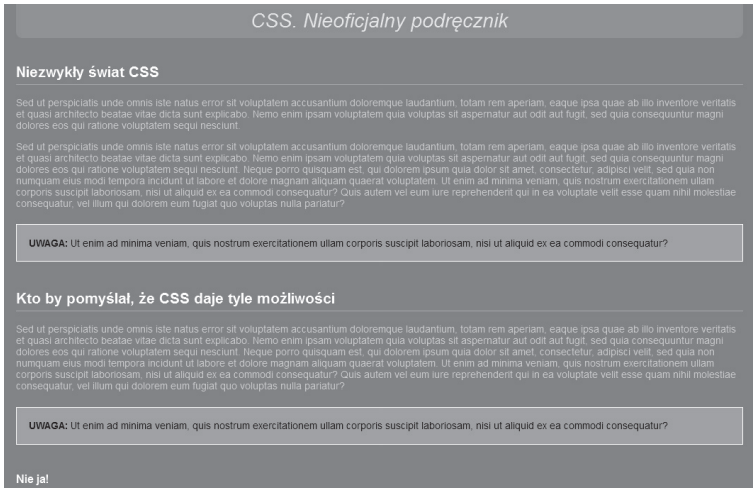
   ```
   <p class="note"><strong>Uwaga:</strong>
   ```

   Uważaj, by *nie* wpisać `class=".note"`. W CSS kropka jest potrzebna do oznaczenia nazwy klasy, w HTML-u jej używanie jest w tym miejscu zabronione. Powtórz te czynności dla drugiego akapitu (znajduje się nad elementem <h3> zawierającym tekst „Nie ja”).

UWAGA Nie ma żadnego powodu, dla którego nie można by było dodać tej klasy do innych znaczników, nie tylko `<p>`. Jeśli zechcielibyśmy dołączyć ją na przykład do znacznika `<h2>`, to kod HTML wyglądałby tak: `<h2 class="note">`.

6. Zapisz stronę i obejrzyj ją w przeglądarce.

Akapit z uwagą jest ładnie wyróżniony (rysunek 3.13).



RYSUNEK 3.13. *Za pomocą selektorów klas można dokonywać bardzo precyzyjnych zmian. Styl klasy zmienił wygląd wybranych akapitów, odróżniając je od pozostałych. Do wyróżnienia widocznej tu notatki użyty został selektor klasy*

UWAGA Jeśli strona nie wygląda tak jak na rysunku 3.13, może być to wina błędu w nazwie właściwości lub jej wartości. W takim przypadku trzeba dokładnie sprawdzić kod z tym z punktów powyżej. Należy się również upewnić, czy wszystkie deklaracje — kombinacje właściwość-wartość — zostały prawidłowo zamknięte znakiem średnika oraz czy na końcu każdego stylu znajduje się nawias klamrowy. Częstą przyczyną tego, że style nie działają tak, jak powinny, są brakujące średniki i zamykające klamry.

Tworzenie selektora potomka

W kodzie strony *selectors_basics.html* zastosowaliśmy klasę `note` do dwóch akapitów. Każdy z nich zaczyna się od pogrubionego słowa „UWAGA" — tak naprawdę znajduje się ono w elemencie `<strong>`, który wszystkie przeglądarki wyświetlają z pogrubieniem. Jak sprawić, aby te pogrubione słowa były pomarańczowe? Można by było utworzyć ogólny styl dla elementu `<strong>`, ale odnosiłby się on do wszystkich tych elementów na stronie, a my chcemy zmienić tylko wygląd elementów `<strong>` znajdujących się w uwagach. Jednym z możliwych rozwiązań jest utworzenie klasy, na przykład `.noteText`, i zastosowanie jej do wybranych elementów. To jednak wymagałoby dużo pracy i jeśli stron byłoby dużo, łatwo można byłoby pominąć niektóre elementy.

Lepszym rozwiązaniem jest użycie selektora potomka wybierającego tylko te elementy `<strong>`, które znajdują się w elementach uwag. Napisanie takiego selektora jest bardzo łatwe.

1. **Przejdź z powrotem do swojego edytora i pliku *selector_basics.html*.** Utwórz nowy pusty wiersz dla stylu selektora potomka.

   Jeśli właśnie wykonałeś poprzedni punkt, kliknij za zamykającym nawiasem klamrowym stylu `.note`, a następnie naciśnij klawisz *Enter*.

2. **Wpisz kod `.note strong {`.**

   Ostatni znacznik w selektorze (`strong`) to element, który ma być bezpośrednio sformatowany. W tym przypadku styl sformatuje tylko te znaczniki `<strong>`, które znajdują się *wewnątrz* elementu przypisanego do odpowiedniej klasy. Nie ma on żadnego wpływu na przykład na znaczniki `<strong>` wewnątrz innych akapitów, list czy nagłówków.

3. **Naciśnij klawisz *Enter*, wpisz `color: #FC6512;` i ponownie naciśnij *Enter*, aby przejść do nowego wiersza. Zakończ styl, wpisując zamykający nawias klamrowy.**

   Ukończony styl powinien wyglądać tak:

```
.note strong {
    color: #FC6512;
}
```

4. **Zapisz stronę i obejrzyj wynik w przeglądarce.**

   Słowo „UWAGA” we wszystkich ramkach uwag powinno mieć pomarańczowy kolor.

Selektory potomka należą do najbardziej przydatnych narzędzi CSS. Profesjonaliści używają ich bardzo często na swoich stronach, aby nie zaśmiecać kodu HTML klasami CSS. W kolejnych rozdziałach dowiesz się jeszcze dużo więcej o tych selektorach i sposobach ich użycia.

Ostatni szlif

Aktualnie tekst zajmuje całą szerokość naszej strony, co można zaobserwować, zmieniając rozmiar okna przeglądarki. Gdy to zrobisz, linijki tekstu będą się rozszerzać wraz z powiększającą się szerokością okna. Jeśli masz duży monitor, to po przekroczeniu pewnego progu tekst stanie się nieczytelny — zbyt długie wiersze po prostu źle się czyta. Na szczęście szerokość treści na stronie można ustawiać, więc da się zapobiec temu niekorzystnemu efektowi.

1. **Wróć do swojego edytora i pliku *selector_basics.html*. Utwórz nowy pusty wiersz na nowy styl.**

   Jeśli wykonałeś poprzednie polecenia, teraz kliknij za zamykającą klamrą stylu `.note strong` i naciśnij klawisz *Enter*.

2. **Dodaj poniższą regułę:**

```
article {
  max-width: 760px;
}
```

   Jest to selektor typu odnoszący się do elementów HTML5 `<article>`, które służą do oznaczania treści głównej na stronie, tzn. artykułów, wpisów itp.

Własność `max-width` określa maksymalną szerokość elementu, czyli w tym przypadku dzięki jej deklaracji element `<article>` nigdy nie będzie szerszy niż 760 pikseli. Zapisz plik i obejrzyj efekt swojej pracy w przeglądarce. Gdy zwiększysz szerokość okna do wartości powyżej 760 pikseli, zauważysz, że treść przestanie się rozszerzać wraz z nim, choć niebieskie tło będzie widoczne na całości.

Gdy natomiast szerokość okna zostanie zmniejszona poniżej 760 pikseli, linijki tekstu również się skrócą. Tak właśnie działa własność `max-width`, która określa szerokość maksymalną, ale nie minimalną. Jest ona niezastąpiona przy projektowaniu stron dla urządzeń o różnych rozmiarach ekranu — komputerów stacjonarnych, laptopów, tabletów i smartfonów. Stanowi też ważny element technik projektowania *responsywnych stron internetowych*, o których jest mowa w rozdziale 17.

Ograniczyliśmy maksymalną szerokość tekstu, więc przydałoby się jeszcze ustawić tę treść na środku ekranu, zamiast pozwalać jej przylegać do lewej krawędzi.

3. Dodaj jeszcze jedną własność do stylu elementu `<article>`; całość powinna wyglądać następująco:

```
article {
  max-width: 760px;
  margin: 0 auto;
}
```

Własność `margin` określa odległość między elementem a otaczającą go treścią. Szerzej o marginesach piszę w rozdziale 7., ale już teraz mogą wyjaśnić, że dodana deklaracja ustawia lewy i prawy margines na `auto`, co dla przeglądarki stanowi sygnał, że ma automatycznie obliczyć, ile pustego miejsca dodać z lewej i prawej strony elementu `<article>`. Gdy szerokość okna przeglądarki przekroczy 760 pikseli, element ten przestanie się rozszerzać i wówczas przeglądarka zacznie po jego obu stronach dodawać pustą przestrzeń, co w efekcie spowoduje jego wyśrodkowanie.

Dla zabawy dodamy jeszcze jeden, trochę bardziej zaawansowany styl — opisany wcześniej selektor brata — formatujący akapit znajdujący się bezpośrednio za pierwszym nagłówkiem na stronie. (To samo można osiągnąć, tworząc klasę i przypisując ją do wybranego akapitu, ale dzięki selektorowi brata nie trzeba nic zmieniać w kodzie HTML).

4. Dodaj jeszcze jedną regułę:

```
h1+p {
  color: rgb(255,255,255);
  font-size: 1.2em;
  line-height: 140%;
}
```

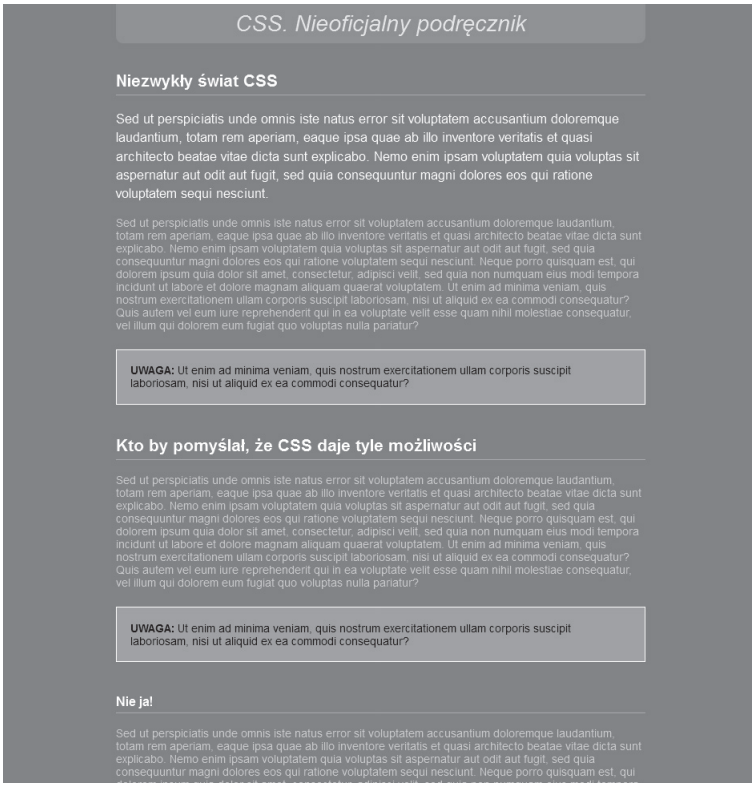
Selektor ten wybiera akapit znajdujący się *bezpośrednio* za elementem `<h1>`, czyli najważniejszym nagłówkiem na stronie. Dotyczy tylko pierwszego takiego akapitu — pozostałe pozostaną bez zmian. Za pomocą tego selektora można w łatwy sposób zastosować indywidualne formatowanie dla wstępnego akapitu, na przykład wyróżnić go wizualnie, aby od razu było wiadomo, że od niego zaczyna się artykuł.

W regule tej zmieniamy kolor i rozmiar czcionki oraz za pomocą własności `line-height` (opisana w rozdziale 6.) ustawiamy odstęp między wierszami tekstu w akapitach (tzw. *leading*).

Teraz tekst w pierwszym akapicie na stronie ma biały kolor oraz nieco większy rozmiar pisma, a ponadto zwiększyły się odstępy między wierszami (rysunek 3.14). Jeśli usuniesz ten akapit z kodu HTML, to jego formatowanie przejmie kolejny, który od tej pory stanie się przylegającym bratem elementu `<h1>`.

Na tym zakończymy krótki przegląd selektorów. Studiując następne rozdziały i zawarte w nich kursy, nabierzesz praktyki w posługiwaniu się tymi i innymi selektorami, tak że będziesz się nimi swobodnie posługiwać. Na razie wystarczy, że wiesz o ich istnieniu i kiedy powinno się ich używać.

UWAGA Ukończoną wersję tej strony można znaleźć w folderze *03_koniec*.



RYSUNEK 3.14. *Strona nabrała całkiem przyjemnego wyglądu. Ustawienie szerokości i definicje własności typograficznych naprawdę uatrakcyjniły nudną i brzydką stronę, od której zaczęliśmy pracę*

Skorowidz

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

Ś

T

U

W

Z

CSS

nieoficjalny podręcznik

CSS jest świetnym narzędziem do budowania profesjonalnych stron internetowych. Opanowanie jego tajników może wydawać się dość trudnym zadaniem, jednak wysiłek ten bardzo się opłaca. CSS wciąż zaskakuje nieoczekiwanymi możliwościami, dalece wykraczającymi poza proste ozdabianie stron WWW. Technologia ta pozwala na tworzenie znakomicie wyglądających witryn o przebogatej funkcjonalności. Co jakiś czas pojawiają się nowe narzędzia i modele, jeszcze bardziej poszerzające warsztat projektanta. CSS jest jedną z tych technik, które wymagają nieustannego uczenia się i zapoznawania z nowościami.

Niniejsza książka to niezwykle wartościowy podręcznik dla projektantów stron o różnym poziomie zaawansowania. Zawarto tu zwięzłe wprowadzenie do języka HTML w zakresie niezbędnym dla każdego, kto chce programować w CSS. Przedstawiono wyczerpujące i dokładne wskazówki tworzenia stron WWW w CSS, wyjaśniając poszczególne niuanse tej technologii. W dobie rozwoju urządzeń mobilnych niezwykle cenne są informacje o sposobach kontroli układów strony, elementach pływających i pozycjonowaniu. Ponadto autor omówił wiele zaawansowanych technik CSS, takich jak systemy siatkowe, model Flexbox, technologia Sass i inne.

W tej książce znajdziesz:

- **zwięzłe wyjaśnienie podstaw HTML i CSS**
- **instrukcje dotyczące programowania kształtów, ramek, cieni, gradientów**
- **omówienie projektowania struktury i układów strony, również dla urządzeń mobilnych**
- **wskazówki dotyczące stosowania takich technik jak elementy pływające i pozycjonowanie**
- **omówienie modelu Flexbox i technologii Sass**

David Sawyer McFarland — utalentowany webdeveloper, nauczyciel i pisarz. Strony internetowe tworzy od 1995 roku, kiedy to zaprojektował e-magazyn dla specjalistów od komunikacji. Uczy projektowania stron WWW w UC Berkeley Graduate School of Journalism, The Center for Electronic Art, Art Institute of Portland i Portland State University. Obecnie jest kierownikiem kadry nauczycielskiej portalu edukacyjnego Treehouse (http://teamtreehouse.com).

Helion
43701 numer katalogowy
księgarnia internetowa
http://helion.pl
zamówienia telefoniczne
0 801 339900
0 601 339900
Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci
Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl
Informatyka w najlepszym wydaniu
ISBN 978-83-283-2289-9
cena: 89,00 zł