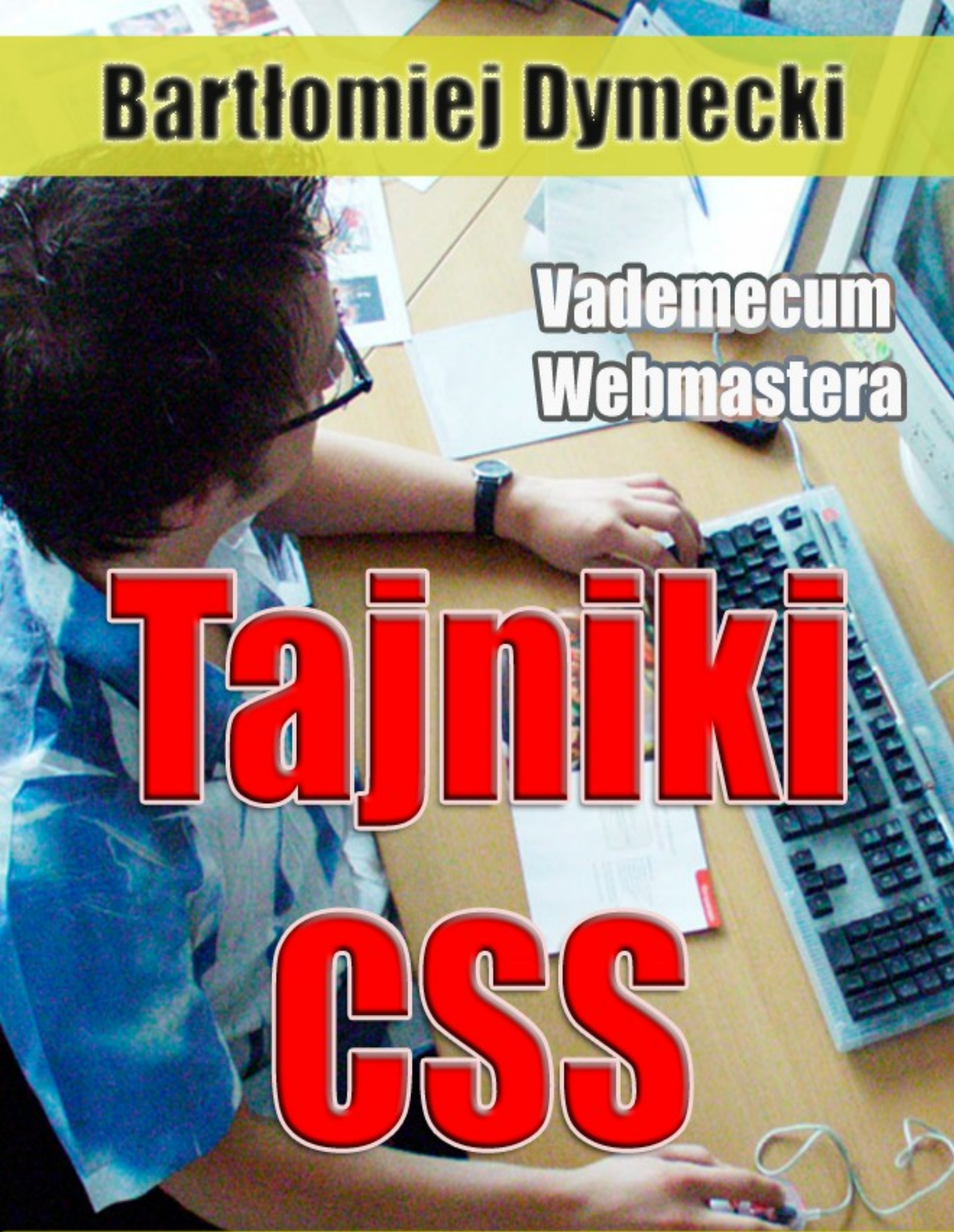
Bartłomiej Dymecki



Vademecum Webmastera

Tajniki CSS

www.EscapeMagazine.pl

Bartłomiej Dymecki

Tajniki CSS

Tajniki CSS

Bartłomiej Dymecki

Skład i łamanie: Patrycja Kierzkowska

Korekta: Anna Matusewicz

Wydanie pierwsze, Jędrzejów 2007

ISBN: 978-83-60320-97-6

Wydawnictwo Escape Magazine
ul. Spokojna 14
28-300 Jędrzejów
http://www.EscapeMagazine.pl

darmowy fragment

Spis treści

Wstęp

Rosnąca popularność internetu sprawia, że coraz więcej osób myśli o stworzeniu własnej strony internetowej. Entuzjaści Sieci marzą o posiadaniu strony domowej, bloga albo nawet większej witryny tematycznej. Mnóstwo internautów w różnym wieku zabiera się do nauki, przede wszystkim języków HTML i CSS.

Niestety, wiele źródeł dostępnych w internecie nie uczy poprawnego używania tych języków. Do niedawna także większość książek karmiła czytelników bajkami o budowaniu stron przy użyciu tabel, zaletach używania ramek i absolutnie nie wspominała o konieczności rozdzielenia struktury od elementów definiujących wygląd witryny. W ciągu kilku ostatnich lat sytuacja pod tym względem nieco poprawiła się, ale duża część internetowych kursów, artykułów i poradników nadal mija się z prawdą.

Najwięcej nieporozumień urosło wokół technologii CSS. Dlatego książka ta nosi tytuł „Tajniki CSS". To, o czym będę pisał, nie jest żadną tajemnicą, ale raczej wiedzą, która nie jest powszechnie znana. Z e-booka skorzystają zarówno osoby początkujące, bo już na starcie nauczą się poprawnego budowania stron, jak i osoby bardziej zaawansowane, które mają okazję zapoznać się z wiedzą trudną do zdobycia jeszcze kilka lat temu i zmienić swoje nawyki.

Wybrałem taką formę publikacji z kilku powodów. Po pierwsze, chciałem stworzyć coś w rodzaju prawdziwego przewodnika po języku - od kwestii podstawowych do bardziej zaawansowanych. Po drugie, ebook jest, przynajmniej dla mnie, o wiele przystępniejszy, niż np. zbiór artykułów na stronie www.

Zajmuję się tworzeniem stron w języku CSS od kilku lat i sporo się przez ten czas nauczyłem. Prowadzę między innymi blog http://www.BelloisNadaje.pl W tej książce będziemy m.in. mówić zarówno o podstawach języka, jak i zaawansowanych selektorach i pseudoklasach. Wyjaśnimy sobie, czym jest kaskada, box model oraz, w jaki sposób można generować treść strony przy pomocy arkuszy stylów. Opiszemy również błędy w obsłudze CSS w najpopularniejszej przeglądarce internetowej, czyli Internet Explorerze. Nie zabraknie oczywiście przykładów praktycznych i interesujących ciekawostek.

Tworząc tę książkę starałem się unikać zbędnego wodolejstwa, pisać o konkretach, a zarazem używać prostego języka. Jeśli uważasz, że coś można w niej ulepszyć albo coś jest dla Ciebie niejasne, to zapraszam do kontaktu ze mną.

Co powinieneś wiedzieć?

Do zrozumienia treści zawartych w tym dziele, niezbędna jest wiedza na temat języka HTML. Jest on na szczęście dużo prostszy, niż sam CSS. Kursy HTML można znaleźć na wielu stronach internetowych, czy w różnych książkach dostępnych w księgarniach.

Wybierz dobrą przeglądarkę

CSS jest językiem dosyć rozbudowanym. Nie każda przeglądarka internetowa rozumie wszystkie jego elementy. Najbardziej blado wypada tutaj popularny (niestety) Internet Explorer. Nie rozumie wielu konstrukcji, a część interpretuje w niepoprawny sposób. Warto używać nowoczesnych przeglądarek - np. Opery (http://www.operapl.net) lub Firefoksa (http://www.firefox.pl). Obie komunikują się z użytkownikiem w języku polskim.

Przykłady

Na potrzeby książki stworzyłem szereg przykładów obrazujących opisywane techniki. Są one umieszczone na stronie wydawnictwa Escape Magazine. W różnych rozdziałach podaję odnośniki kierujące bezpośrednio do przykładów.

Możesz także przejrzeć ich spis: **(link w pełnym wydaniu)**

Jeżeli chcesz pobrać wszystkie przykłady, to są one spakowane:

(link w pełnym wydaniu)

Rozdział 1

Wprowadzenie do CSS

Skrót CSS pochodzi od angielskiej nazwy *Cascading Style Sheets*, co tłumaczy się jako *kaskadowe arkusze stylów*. Język CSS łączy się z językiem HTML i służy do określania **wyglądu** stron internetowych.

Podstawowym i najważniejszym powodem używania arkuszy stylów jest możliwość oddzielenia elementów odpowiedzialnych za wygląd strony od jej struktury. Inaczej - warstwy prezentacji od warstwy informacji. Jednym z celów przy pisaniu kodu CSS jest umieszczenie w nim **wszelkich** informacji o wyglądzie witryny. Mówiąc prościej: na stronie w języku HTML zapisujesz jakiś tekst,za pomocą języka CSS opisujesz wygląd npkolor tekstu, krój czcionek, itd.

Ogromnie ułatwia to późniejszą edycję strony. Jeśli chcę zmodyfikować jakieś szczegóły wyglądu, to najczęściej wystarczy, że zmodyfikuję kilka reguł CSS. W pewnych przypadkach ingerencja w kod HTML także jest konieczna, ale ma ona wówczas bardziej ograniczony charakter i jest o wiele prostsza do przeprowadzenia.

W tym rozdziale omówimy sobie dokładnie zasady działania CSS. Zaczniemy od metod dołączania stylów do kodu HTML.

Jak połączyć HTML i CSS?

Kod CSS możemy dołączyć do dokumentu HTML na trzy sposoby. Pierwszy z nich, to umieszczenie kodu bezpośrednio przy znaczniku:

```
<p style="..."></p>
```

Jest to oczywiście sposób **zły**, ponieważ nie oddzielimy wyglądu od struktury, co poskutkuje niezwykle zagmatwanym i trudnym do odczytania kodem.

Drugi sposób, to umieszczenie kodu w sekcji HEAD dokumentu:

```
<head>
   <style type="text/css">
         ...
   </style>
</head>
```

Jest to metoda dobrze sprawdzająca się w przypadku niedużych i mało skomplikowanych stron.

Trzecia metoda, idealna w większości przypadków, polega na umieszczeniu kodu CSS w osobnym pliku i dołączeniu go do dokumentu przy pomocy znacznika LINK:

```
<head>
   <link rel="stylesheet" href="style.css" type="text/css" />
</head>
```

Omawianie szczegółów dotyczących tego znacznika nie wchodzi w zakres tematyczny tej książki. Informacje na ten temat można znaleźć w Sieci, na przykład na tej stronie: http://grabun.com/teksty/odnosniki-w-sekcji-head

Aby ostatnia metoda zadziała, musisz oczywiście utworzyć plik o nazwie: *style.css*. (*style*, to najczęściej nadawana nazwa i takiej też użyłem w przykładzie; oczywiście można zastosować inną). Arkusz stylów zawsze powinien posiadać rozszerzenie **.css*.

Nic nie stoi także na przeszkodzie, aby do jednej strony dołączyć nawet kilka osobnych plików ze stylami. W niektórych przypadkach byłoby to nawet wskazane. Podam prosty przykład z własnego doświadczenia. Prowadzę blog poświęcony Sieci i webmasteringowi. Dla jego poprzedniej wersji stworzyłem dwa osobne pliki ze stylami.

Pierwszy jest ogólny - dotyczy zarówno strony głównej, jak i wszystkich podstron. Drugi wczytywany jest tylko na podstronach z konkretnym wpisem, ponieważ zawiera m.in. style odnoszące się do komentarzy i formularza komentowania, które nie są potrzebne na stronie głównej. Dzięki takiemu rozwiązaniu strona główna działa szybciej. Różnica nie jest być może kolosalna, ale kilka drobnych zmian zsumowanych ze sobą może mieć już większe znaczenie. Dlatego wszędzie gdzie się tylko da, należy szukać metod optymalizacji strony - grosik do grosika.

Tworzenie reguł

Najważniejszym elementem stylów CSS jest *reguła*. Składa się z *selektora* oraz zestawów *właściwości* i *wartości*. Obrazuje to prosty schemat:

```
selektor {
   właściwość1: wartość1;
   właściwość2: wartość2;
   właściwość3: wartość3
}
```

Selektor określa, do jakich elementów odnosi się dana reguła. Pary właściwości i wartości zamknięte są parze nawiasów klamrowych. Po każdej wartości, występuje średnik. W przypadku ostatniej nie jest on konieczny. Oto przykład:

```
p {
   color: black;
   font-size: 1.2em
}
```

W tym przypadku selektorem jest znacznik (inaczej *tag*) P, więc reguła definiuje wygląd akapitu oznaczanego w języku HTML znacznikiem P. Pierwsza para właściwości i wartości ustala czarny kolor tekstu, a druga wielkość liter. W tej chwili szczegóły nie są dla nas ważne, chodzi o zrozumienie zasady.

Posiłkując się tym schematem:

```
właściwość: wartość;
```

możemy powiedzieć, że właściwością jest słowo *color* definiujące kolor tekstu. Właściwość ta przyjmuje wartość *black*, czyli *czarny*. Tak więc kolor tekstu w akapicie będzie czarny.

Definiując reguły w CSS nie trzeba robić żadnych odstępów, czy wcięć, lecz jest to wskazane, ponieważ pozwala zachować przejrzystość kodu.

Mam nadzieję, że to wszystko jest dla Ciebie proste i klarowne. Dokładniej o dostępnych właściwościach i wartościach powiemy sobie w jednym z rozdziałów. Jeśli masz jakieś wątpliwości, to wróć do początku tego rozdziału lub czytaj dalej, a niebawem zostaną one rozwiane.

Komentarze

Obok reguł można umieścić własne komentarze. Jest to szczególnie przydatne w przypadku bardziej rozbudowanych arkuszy. Komentarze wstawiamy w niezwykle prosty sposób:

```
/* treść komentarza */
```

Tutaj chyba nie potrzeby dłuższego tłumaczenia tego zagadnienia. Stosuj komentarze tam, gdzie po pewnym czasie możesz mieć problemy ze zrozumieniem znaczenia kodu. Komentarze są również dobre do dzielenia arkuszy stylów na kilka sekcji, o czym dokładniej opowiem pod sam koniec książki.

Wkraczamy w świat selektorów

Dobrze. Wiesz już, czym są selektory. Teraz porozmawiamy o różnych typach selektorów. W tym rozdziale omówimy tylko najprostsze z nich. Aby poznać te bardziej zaawansowane, możesz sięgnąć do rozdziału 8.

Selektor elementu. Ten rodzaj wystąpił w naszym wcześniejszym przykładzie. Odnosi się on po prostu do danego znacznika HTML, jak P, H1, czy BLOCKQUOTE. Spójrzmy na to jeszcze raz:

```
blockquote {
   ...
}
```

Selektor identyfikatora. Jak zapewne wiesz, elementom HTML można nadawać unikalne identyfikatory, np.:

```
<form id="zamowienie">
```

Dany identyfikator może być przypisany tylko i wyłączanie do jednego znacznika, czyli może wystapić tylko raz w dokumencie HTML. Jeśli mamy w kodzie formularz z identyfikatorem *zamowienie*, to możemy się do niego odwołać w regule CSS:

```
#zamowienie {
   ...
}
```

Tak więc, aby reguła odnosiła się tylko i wyłącznie do konkretnego elementu, musimy stworzyć selektor składający się z dowolnej nazwy poprzedzonej znakiem #. Następnie przy pomocy atrybut *id* podpinamy ją do dowolnego znacznika.

Selektor klasy. Selektora klasy możemy użyć wtedy, gdy chcemy stworzyć regułę odnoszącą się do **kilku** wybranych elementów. Podam gotowy przykład. Załóżmy, że na Twojej stronie jest kilka ciekawych tekstów i chciałbyś od czasu do czasu wyróżnić w nich jakieś zdanie przez zmianę jego koloru.

Dawniej sądzono, że efekt taki można osiągnąć przez użycie znacznika FONT. My jednak wiemy, że należy oddzielać strukturę od wyglądu, więc zrobimy to w CSS:

```
.niebieski {
   color: blue
}
```

Jak widzisz, nazwy klas są zawsze poprzedzone kropką. Dzięki znacznikowi SPAN możemy łatwo użyć naszej klasy w kodzie HTML, posługując się właściwością *class*:

```
<p>
   Normalny tekst w akapicie.
    <span class="niebieski">
A tu tekst wyróżniony na niebiesko.
</span>
    I znów tekst normalny.
</p>
```

Łączenie selektorów. Selektory możemy również łączyć ze sobą. Załóżmy, że chcemy, aby tekst umieszczony w kilku znacznikach miał zawsze czerwony kolor:

```
h1, p, a {
   color: red
}
```

Jeżeli więc chcesz przypisać daną właściwość do kilku elementów, to możesz użyć kilku selektorów i oddzielić je przecinkami. Bardziej zaawansowany przykład:

```
#pierwszy, .niebieski, p {
   ...
}
```

Nie ma znaczenia, czy są to selektory identyfikatorów, klas czy też selektory elementów.

Selektor gwiazdki. Odnosi się on do absolutnie wszystkich znaczników:

```
* {
   ...
}
```

Selektor gwiazdki wpływa zarówno na akapity, jak i nagłówki. Na wszystkie elementy strony.

Czym jest kaskada?

Wiedza na temat kaskady jest niezwykle istotna dla pełnego zrozumienia działania arkuszy stylów, które nie bez powodu noszą miano **kaskadowych**.

Generalnie chodzi o to, że różne reguły CSS mają różną ważność - niektóre są ważniejsze i mogą nadpisywać inne, a niektóre są mniej ważne. Style, jak już przecież mówiliśmy, możemy dołączać do strony na kilka różnych sposobów. Stopień ich ważności przedstawia się następująco:

1. Style bezpośrednio przy znaczniku.
2. Style w sekcji HEAD dokumentu.
3. Style w zewnętrznym pliku.

Oznacza to, że reguły CSS znajdujące się w zewnętrznym pliku mogą być **nadpisane** przez reguły znajdujące się wewnątrz pliku HTML, ale nie odwrotnie.

Drugim elementem wpływającym na ważność reguł jest rodzaj selektora. Zasada jest prosta - im dokładniejszy selektor, tym ważniejszą tworzy regułę. Najmniej precyzyjny, a przy tym najmniej istotny jest selektor * - odnoszący się do wszystkich tagów. Następne w kolejności są selektory elementu, jak *div*, czy *ul*. Po nich występują selektory atrybutów, czy pseudoklas, o których będziemy mówić w dalszej części książki. Najważniejszym selektorem jest selektor identyfikatora.

Może jednak zaistnieć sytuacja w której chcielibyśmy, aby jedna reguła była ważniejsza, niż wynikałoby to z zasad kaskady. Możemy sprawić, że dana reguła będzie najważniejsza przez wykorzystanie specjalnego atrybutu *!important*. Spójrzmy na pewien przykład kodu HTML:

```
<ul id="lista">
   <li>jeden</li>
   <li>dwa</li>
   <li>trzy</li>
</ul>
```

Mamy tu prostą listę z nadanym identyfikatorem *#lista*. Dodajmy do tego kod CSS:

```
#lista {
   color: red
}
ul {
   color: blue !important
}
```

Selektor identyfikatora jest ważniejszy, niż selektor elementu, więc tekst w przykładowej liście powinien teoretycznie posiadać czerwony kolor. Jednak atrybut *!important* nadał drugiej regule większą ważność, więc będzie on miał kolor niebieski.

Podsumujmy ważność poznanych dotąd selektorów. Przedstawmy je w kolejności od najmniej znaczącego:

selektor gwiazdki - *

selektor elementu - *p*, *div*, *h1*, *form*

selektor klasy - *.niebieski*, *.trzeci*, *.nowy*

selektor identyfikatora - *#naglowek*, *#menu*, *#wstep*

zwiększanie ważności - *!important*

Rozdział 6

Zaawansowane selektory

Jeśli dotrwałeś do tego momentu, to znaczy, że już całkiem sporo wiesz na temat CSS. Pora więc przejść na wyższy poziom. W tym rozdziale zajmiemy się różnymi zaawansowanymi selektorami. Przed rozpoczęciem lektury warto naprawdę dobrze przyswoić sobie część rozdziału trzeciego mówiącą o podstawowych selektorach. Wierzę, że już to zrobiłeś, ale jeśli nie, to warto zrobić sobie małą powtórkę.

Selektory to potężne narzędzie. Jeżeli nauczysz się dobrze nimi posługiwać, to naprawdę sporo osiągniesz. Jednak równie dobrze mogą być one przyczyną Twoich frustracji, gdy będziesz niedokładny, albo nie do końca zrozumiesz ich działanie. Dlatego skup się mocno i postaraj przyswoić tę wiedzę.

Selektor potomka

Używając tego selektora możemy nadać styl tylko elementom znajdującym się wewnątrz innego elementu. Użycie jest bardzo proste:

```
p a {
   color: red
}
```

Powyższa reguła pozwala nadać inny styl wszystkim odnośnikom znajdującym się wewnątrz akapitów. Popatrzmy na taki kod HTML:

```
<p>
   <a href="">link1</a>
   <a href="">link2</a>
</p>
   <a href="">link3</a>
```

Zastosowany przez nas styl zmieni kolor tekstu dla *link1* i *link2*, ale nie dla *link3*, ponieważ znajduje się on poza akapitem.

Selektory dziecka

Tego selektora Internet Explorer 6 niestety nie rozpoznaje, ale na szczęście zmieniło się to w jego nowej, siódmej wersji, która ukazała się pod koniec 2006 roku.

Selektor dziecka jest podobny do selektora potomka. Jednak w odróżnieniu od niego zadziała tylko wtedy, gdy element zawarty jest **bezpośrednio** w elemencie nadrzędnym. Weźmy taki kod CSS z selektorem potomka:

```
#tresc p {
   ...
}
```

Zadziała on w kodzie HTML:

```
<div id="#tresc">
   <p>...</p>
</div>
```

Zadziała również w takim kodzie:

```
<div id="tresc">
   <div class="jakas_klasa">
         <p>...</p>
   </div>
</div>
```

Dla selektora potomka nie ma znaczenia, czy element znajduje się w nim bezpośrednio, czy poprzedzają go inne elementy.

Zamieńmy teraz selektor sąsiada na selektor dziecka:

```
#tresc > p {
   ...
}
```

W przypadku takiego selektora styl zadziała tylko dla znacznika P znajdującego się **bezpośrednio** w DIV-ie *#tresc*. Zadziała dobrze dla takiego kodu HTML:

```
<div id="tresc">
   <p>...</p>
</div>
```

Ale nie dla takiego:

```
<div id="tresc">
   <blockquote>
         <p>...</p>
   </blockquote>
</div>
```

Selektor sąsiada

Selektor sąsiada również nie jest obsługiwany przez IE6, a tylko IE7. Składa się on tak naprawdę z dwóch selektorów. Odnosi się do sytuacji, gdy w kodzie HTML drugi selektor występuje bezpośrednio za tym pierwszym. Tak się go konstruuje:

```
h2 + p {
   ...
}
```

W tym przypadku akapit występujący bezpośrednio po nagłówku drugiego poziomu otrzymałby specjalny styl:

```
<h2>Nagłówek</h2>
<p>...</p>
```

Ale jeśli poprzedzałby go inny znacznik, to selektor nie mógłby zadziałać:

```
<h2>Nagłówek</h2>
<h3>...</h3>
<p>...</p>
```

Selektor sąsiada można wykorzystać np. do wyróżnienia pierwszego akapitu wewnątrz artykułu.

Selektory pseudoklas

Na początku przyjrzymy się konstrukcji pseudoklasy:

```
selektor:pseudoklasa {
   ...
}
```

Widzisz już tę zasadę? Najpierw występuje normalny selektor, potem dwukropek, a następnie nazwa pseudoklasy. Czym są pseudoklasy? Najłatwiej jest to wyjaśnić omawiając wszystkie dostępne pseudoklasy:

:link - odnosi się do nieodwiedzonych odnośników,

:visited - odnosi się do odwiedzonych odnośników,

:hover - działa wtedy, gdy nad elementem znajduje się wskaźnik myszy,

:focus - definiuje właściwości elementu aktualnie używanego np. pole formularza podczas wpisywania danych. Nie działa w IE 6,

:active - element jest aktywny, np. został kliknięty,

:first-child - odnosi się tylko do pierwszego elementu znajdującego się wewnątrz selektora, np. pierwszego akapitu. Nie działa w IE 6,

:lang - pozwala nadać styl elementom, które mają nadany atrybut *lang*. Niestety również nie działa w przeglądarce IE 6,

Łatwe? No to zajmijmy się przykładami praktycznego zastosowania pseudoselektorów.

Najbardziej typowy przykład, to style dla odnośników, nad którymi znajduje sie kursor myszy:

```
a {
   color: darkred;
   text-decoration: none
}
```

```
a:hover {
   color: red;
   text-decoration: none
}
```

W wyniku zastosowania tego stylu, hiperłącze będzie miało kolor ciemnoczerwony bez żadnych zdobień. Gdy znajdzie się nad nim kursor myszy, przybierze kolor czerwony.

Właściwość *text-decoration* dwukrotnie ustawiam na *none*, ponieważ nie chcę żadnych podkreśleń. Różne przeglądarki mają skłonności do ustawiania wartości tej właściwości według własnego *widzi-mi-się*, więc dobrze jest zrobić to samodzielnie zarówno dla normalnego i hiperłącza, jak i dla *:hover*.

Spróbujmy teraz zrobić coś z pseudoklasą *:lang*. Jak zapewne wiesz, znacznik *BLOCKQUOTE* oznacza dłuższy cytat. Nadajmy inny styl cytatom w języku angielskim.

```
blockquote:lang(en) {
   ...
}
```

Styl ten będzie mieć zastosowanie dla kodu HTML:

```
<blockquote lang=”en”>
   ...
</blockquote>
```

Zastanówmy się jeszcze nad wykorzystaniem *:first-child*. Możemy na przykład nadać styl pierwszemu elementowi listy UL:

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

Oprócz wyżej wspomnianych, istnieje sporo innych pseudoklas, których jednak w tym momencie raczej się nie stosuje z powodu braku ich obsługi w różnych przeglądarkach. Mam tu na myśli na przykład: *:last-child*. *:root*, *:first-of-type*, *:empty*, *:only-child* i wiele innych. Jeżeli będą one obsługiwane chociażby przez Firefox i Operę, to ich szczegółowy opis z całą pewnością zostanie tutaj dodany.

Selektory pseudoelementów

Bardzo podobne do selektorów pseudoklas są selektory pseudoelementów. Mają one identyczną składnie. Dostępne są dwa tego rodzaju selektory:

:first-line - pozwala zmodyfikować pierwszą linię tekstu znajdującą się w danym elemencie, np. akapicie.

:firts-letter - ten selektor pozwala z kolei zmodyfikować pierwszą literę danego tekstu.

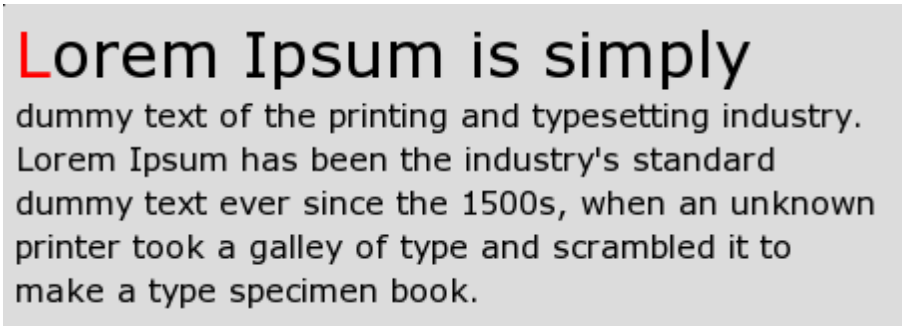
Zobaczmy, jak to wygląda w praktyce. Mamy prosty kod HTML:

```
<p>Lorem Ipsum is simply dummy text of the printing and typesetting
industry. Lorem Ipsum has been the industry's standard dummy text ever
since the 1500s, when an unknown printer took a galley of type and
scrambled it to make a type specimen book. </p>
```

Przykładowy tekst został pobrany ze strony http://www.lipsum.com. Teraz spójrz na CSS:

```
p:first-line {
   font-size: 2em
}
p:first-letter {
   color: red
}
```

Efekt:



Ponieważ selektor dotyczy pierwszej linii, to gdy zwiększymy rozmiar okna przeglądarki...

...zwiększy się również ilość tekstu objętego stylem.

Selektory atrybutów

Na pewno wiesz, że znacznikom języka HTML można nadawać różne atrybuty (jak TITLE, ALT, HREF, NAME i wiele innych). Do atrybutów tych możemy się odwoływać również przy pomocy reguł języka CSS. Dzięki temu możemy nadać styl **tylko tym elementom, którym nadano określone atrybuty**. Daje to nam, jako webmasterom, ogromne możliwości. W ogóle nie rozumie ich Internet Explorer 6, ale do wersji 7 została dodana ich obsługa.

Najpierw przeanalizujmy wszystkie dostępne selektory atrybutów:

selektor[atrybut] - aby selektor zadziałał, wystarczy sama obecność atrybutu. Przykład:

```
a[title] {
   color: red
}
```

Znacznik A ma nadany atrybut TITLE, więc zostanie mu zaaplikowany styl zmieniający jego kolor na czerwony.

selektor[atrybut~="wartość"] - selektor sprawdza, czy dane słowo występuje wewnątrz atrybutu. Na przykład:

```
a[title~="nowy"] {
   ...
}
```

Powyższy selektor zadziała tylko wtedy, gdy słowo *nowy* wystąpi w atrybucie TITLE dla znacznika A. Wybrane słowo musi być oddzielone spacjami od innych znaków. Zadziała więc w takich przypadkach:

```
<a title="nowy" href="">...</a>
<a title=" super nowy" href="">...</a>
<a title="super nowy motor" href="">...</a>
```

Ale nie zadziała w takich:

```
<a title="super motor” href="">...</a>
<a title="supernowy” href="">...</a>
```

selektor[atrybut="wartość"] - sprawdza, czy zawartość atrybutu jest **dokładnie taka sama**, jak podanej wartości. Spójrz na przykład podobny do poprzedniego:

```
a[title="Mój nowy tytuł"] {
   ...
}
```

Odnośnik musi mieć nadany tytuł składający się tylko i wyłącznie ze sformułowania *Mój nowy tytuł*, aby selektor mógł zadziałać. Oto JEDYNY przypadek w którym zadziała:

```
<a title="Mój nowy tytuł" href="">...</a>
```

selektor[atrybut^="wartość"] - ten i następne selektory zostały zaczerpnięte z języka CSS w wersji 3 (nie jest on jeszcze ukończony). Ten odnosi się do atrybutów, które **rozpoczynają się** określonym ciągiem znaków. Zobaczmy, jak to działa:

```
a[title^="Mój"] {
   ...
}
```

Powyższy selektor zadziała tylko dla znacznika A którego atrybut TITLE rozpoczyna się słowem *Mój*. Zadziała w takich sytuacjach:

```
<a title="Mój motor" href="">...</a>
<a title="Mój nowy motor" href="">...</a>
```

Ale nie w takiej:

```
<a title="Nowy motor" href=""></a>
```

selektor[atrybut$="wartość"] - selektor podobny do poprzedniego, ale odnosi się do atrybutów, których wartość **kończy się** danym ciągiem znaków. Np.:

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

Powyższa reguła zadziała dla odnośników, których atrybut *href* kończy się znakami *pdf*. Dzięki temu możesz nadać inny styl linkom do plików PDF! Możesz w ten sposób nadać styl linkom do dowolnego rodzaju plików. Możesz nawet automatycznie wstawić przy nich odpowiedni obrazek... ale o tym opowiem w rozdziale 13, mówiącym o generowaniu treści przy użyciu CSS. Jeśli jesteś bardzo niecierpliwy, to możesz zajrzeć tam od razu :-)

selektor[atrybut="wartość"]* - jest to selektor zbliżony do selektora tworzonego przy pomocy ~=, ale w jego przypadków nie ma znaczenia, przy pomocy jakich znaków dane słowo zostało oddzielone od innych. Tak więc reguła podobna do jednej z poprzednich:

```
a[title*="nowy"] {
   ...
}
```

zadziała w takich przypadkach:

```
<a title=" super nowy" href="">...</a>
<a title="super nowy motor" href="">...</a>
```

Ale również w takich:

```
<a title=" supernowy" href="">...</a>
<a title=" nowymotor" href="">...</a>
```

Dla tej reguły liczy się tylko to, że zbiór czterech liter tworzący słowo *nowy* w ogóle występuje w atrybucie TITLE.

Zobacz pełną wersję ebooka

Tajniki CSS

http://www.escapemagazine.pl/297324-tajniki-css