Biblia webmastera!

CSS3

nieoficjalny podręcznik

Wydanie III

David Sawyer McFarland

O'REILLY

Helion

Tytuł oryginału: CSS3: The Missing Manual, 3rd Edition

Tłumaczenie: Łukasz Piwko

ISBN: 978-83-246-7317-9

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/css3n3
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/css3n3.zip

Printed in Poland.

Spis treści

7

ROZDZIAŁ

Marginesy, dopełnienie i obramowanie

Wygląd każdego znacznika HTML w przeglądarce jest zdeterminowany przez mnóstwo właściwości. Niektóre z nich — jak obramowanie i kolory tła — widać już na pierwszy rzut oka. Inne zaś są niewidoczne — jak dopełnienie i marginesy, które dodają nieco pustej przestrzeni z jednej lub kilku stron znacznika. Dzięki wiedzy o sposobie funkcjonowania tych właściwości można tworzyć przyciągające wzrok kolumny czy dekoracyjne paski boczne i kontrolować wolną przestrzeń (ang. *white space*) wokół nich. To z kolei pomaga uzyskać uporządkowany i lekko wyglądający projekt, który daje wrażenie profesjonalizmu.

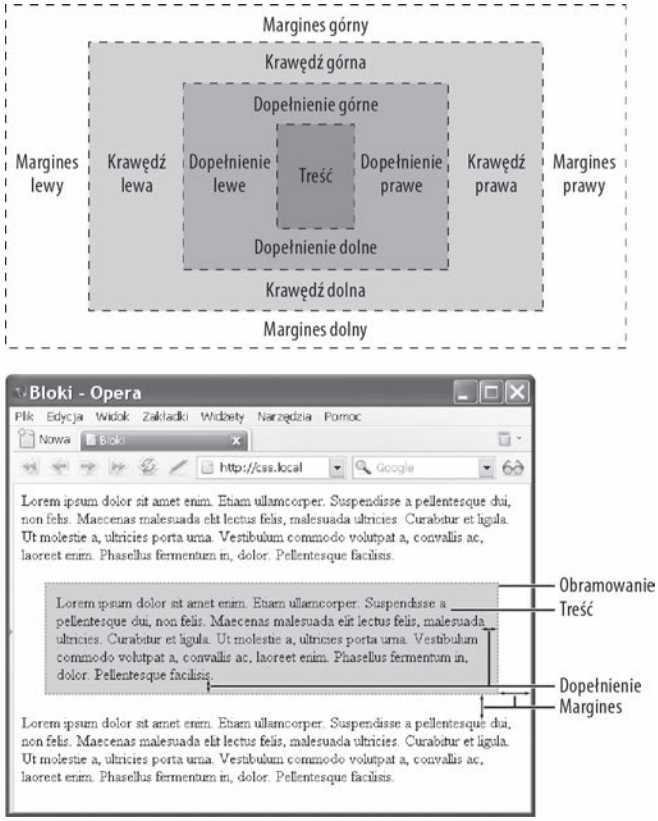
Wszystkie właściwości opisane w tym rozdziale wzięte razem składają się na jedną z najważniejszych koncepcji w CSS — model polowy (zwany też *modelem blokowym* lub *pudełkowym*).

Istota modelu polowego

Mówiąc o akapicie lub nagłówku, zawsze mamy na myśli litery, słowa i zdania. Prawdopodobnie więc mówiąc o znaczniku `<img />`, mamy na myśli zdjęcie, logo albo jeszcze jakiś inny rodzaj obrazu. Przeglądarka jednak traktuje je (i wszystkie inne znaczniki) inaczej — jako małe pola. Dla przeglądarki każdy znacznik to pole, na którym coś się znajduje — obraz, tekst lub inne znaczniki zawierające jeszcze coś innego, jak widać na rysunku 7.1.

Treść jest otoczona różnymi właściwościami, które składają się na pole:

- *Dopełnienie* (`padding`) to przestrzeń pomiędzy treścią a jej obramowaniem. Jest to odstęp znajdujący się pomiędzy zdjęciem a jego obramowaniem.
- *Obramowanie* (`border`) to linia rysowana wokół wszystkich krawędzi pola. Można ją zastosować do dowolnej liczby krawędzi pola w wybranym układzie.



Rysunek 7.1. *Model polowy CSS przewiduje, że w jednym znaczniku może znajdować się właściwa treść (np. kilka zdań) plus dopełnienie, obramowanie i marginesy. Obszar w obrębie ramek, który zawiera treść i dopełnienie, może mieć również jakiś kolor. W rzeczywistości kolor tła jest nakładany pod obramowaniem, dzięki czemu jeśli obramowanie jest przerywane, to w przerwach pomiędzy kropkami lub kreskami go widać*

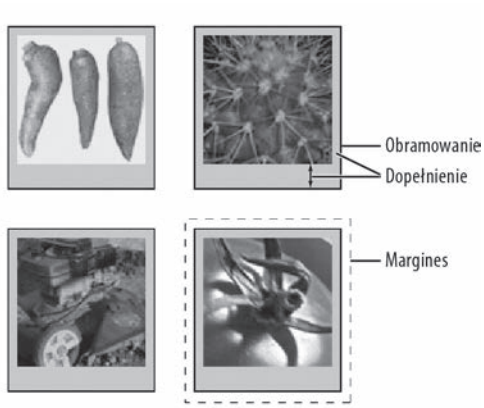
- *Kolor tła* (`background-color`) wypełnia powierzchnię wewnątrz ramek, włącznie z obszarem zajmowanym przez dopełnienie.
- *Margines* (`margin`) to przestrzeń oddzielająca od siebie poszczególne znaczniki. Na przykład marginesem jest powszechnie spotykany odstęp między górną a dolną krawędzią akapitów na stronach.

Właściwości te można stosować w dowolnych kombinacjach do określonych znaczników. Znacznikowi można ustawić tylko margines lub nadać mu obramowanie, marginesy *i* dopełnienie. Nic nie stoi na przeszkodzie, aby zastosować tylko margines, bez dopełnienia itd. Jeśli nie zdefiniujemy którejś z tych właściwości, to przeglądarka zastosuje swoje ustawienia domyślne, które nie muszą nam się podobać. Przykładowo, mimo że przeglądarki z reguły nie stosują dopełnienia ani obramowania do znaczników domyślnie, niektóre znaczniki, takie jak nagłówki i akapity, mają ustawione domyślne wartości marginesów górnego i dolnego.

Uwaga: Każda przeglądarka ma inne domyślne ustawienia dopełnienia i marginesów dla poszczególnych znaczników i dlatego najlepiej jest wyzerować je wszystkie, a następnie zdefiniować od nowa (więcej na ten temat znajdziesz w rozdziale 17.). Innymi słowy, należy za pomocą zestawu prostych stylów — zwanych resetem CSS — całkowicie usunąć marginesy i dopełnienie z elementów. Dzięki temu po zdefiniowaniu własnych ustawień można mieć pewność, że nasze strony w każdej przeglądarce będą wyglądać jednakowo.

Marginesy i dopełnienie

Zarówno marginesy, jak i dopełnienie tworzą przestrzeń wokół treści strony. Stosuje się je w celu oddzielenia od siebie poszczególnych jej elementów — na przykład menu nawigacyjnego po lewej stronie od głównej treści strony po prawej — lub do ustawienia odstępu pomiędzy krawędzią a treścią. Możemy na przykład zechcieć odsunąć obramowanie od krawędzi zdjęć (rysunek 7.2).



Rysunek 7.2. *Każde ze zdjęć na tym rysunku ma ustawiony dziesięciopikselowy margines, a więc przestrzeń między nimi wynosi 20 pikseli. Dopełnienie oddziela obrazy od ich ramek, ukazując szare tło. Właściwości ramek, marginesów i dopełnienia można ustawiać niezależnie dla każdej krawędzi. Zwróć uwagę na powiększone dopełnienie na dole każdego zdjęcia*

Marginesy i dopełnienie funkcjonują w podobny sposób i jeśli nie zostanie zdefiniowane obramowanie albo jakiś kolor tła, nie da się rozróżnić, czy dany odstęp to margines, czy dopełnienie. Jeśli jednak wokół elementu jest obramowanie albo ma on tło, to różnica pomiędzy tymi dwiema właściwościami jest dobrze widoczna. Dopełnienie to odstęp pomiędzy treścią a obramowaniem, zapobiega ono wrażeniu stłoczenia treści w polu. W obszarze dopełnienia widoczne jest tło, a więc mimo że obszar ten jest pusty, może mieć nawet jakiś kolor lub wzór w tle. Marginesy natomiast tworzą odstęp pomiędzy elementami, nadając stronie wrażenie lekkości.

Margines i dopełnienie można ustawiać z każdej strony oddzielnie. Właściwości służące do kontrolowania marginesów to: `margin-top` (margines górny), `margin-right` (margines prawy), `margin-bottom` (margines dolny) i `margin-left` (margines lewy). Dla dopełnienia są cztery podobne właściwości: `padding-top` (dopełnienie górne), `padding-right` (dopełnienie prawe), `padding-bottom` (dopełnienie dolne) i `padding-left` (dopełnienie lewe). Rozmiar marginesów i dopełnienia można określać za pomocą wszystkich jednostek dostępnych w CSS:

```
margin-right: 20px;
padding-top: 3em;
margin-left: 10%;
```

Powszechnie stosowane są piksele i jednostki em, które w tym przypadku funkcjonują tak samo jak z tekstem. Dwudziestopikselowy margines tworzy odstęp o szerokości 20 pikseli, a dopełnienie o szerokości 3 em tworzy odstęp o szerokości równej trzykrotnemu rozmiarowi tekstu stylizowanego elementu. Można też używać wartości procentowych, ale jest to ryzykowne (szczegóły w ramce poniżej).

PORADNIK INŻYNIERA

Marginesy i dopełnienie a wartości procentowe

Jeśli użyjemy jednostek procentowych, to przeglądarka będzie obliczać szerokość odstępu na podstawie *szerokości elementu rodzica*. W przypadku prostych stron elementem tym jest ciało strony, które wypełnia całe okno przeglądarki. W takiej sytuacji wartości procentowe obliczane są na podstawie szerokości okna przeglądarki w określonym czasie. Załóżmy, że okno ma 760 pikseli szerokości, a zatem dziecięcioprocentowy margines oznaczałby odstęp o szerokości 76 pikseli. Ale jeśli zmienimy rozmiar okna, to wartość ta ulegnie zmianie. Zmiana szerokości na 600 pikseli spowoduje zmniejszenie marginesu do 60 pikseli (10 procent z 600).

Jednak element rodzica nie zawsze ma szerokość równą szerokości okna przeglądarki. Jak zobaczymy w następnych rozdziałach, w których będziemy tworzyć bardziej wyrafinowane układy stron, można wprowadzać nowe elementy pomagające w organizacji strony. Można przykładowo wstawić dodatkowy znacznik `<div>` grupujący powiązaną ze sobą treść w jednym pasku bocznym (przykład tego można obejrzeć w kursie w rozdziale 8.). Niech ma on szerokość 300 pikseli. Znacznik ten jest rodzicem elementów w nim zawartych, a więc jeśli znajdzie się w nim znacznik o marginesie ustawionym na 10 procent, to rzeczywista jego szerokość wyniesie 30 pikseli.

Aby jeszcze bardziej utrudnić nam życie, marginesy górny i dolny wyrażone w procentach również obliczane są na podstawie szerokości, a nie wysokości elementu nadrzędnego. A więc margines górny znacznika ustawiony na 20 procent będzie miał szerokość równą 20 procentom szerokości elementu go zawierającego.

Wskazówka: Aby całkiem pozbyć się marginesu lub dopełnienia, należy zastosować wartość `0` (np. `margin-top: 0` lub `padding-bottom: 0`). Aby usunąć odstęp oddzielający od elementów strony wszystkie cztery krawędzie okna przeglądarki — tak żeby na przykład baner lub logo przylegały bezpośrednio do krawędzi okna — należy marginesy i dopełnienie znacznika `<body>` ustawić na zero: `margin: 0;` i `padding:0;`.

Zapis skrótowy marginesów i dopełnienia

Często chcemy ustawić marginesy lub dopełnienie z wszystkich czterech stron elementu, ale wpisywanie wszystkich czterech właściwości (`margin-right`, `margin-left` itd.) dla każdego stylu jest bardzo pracochłonne. Jest na to rada: w takich sytuacjach można stosować zapis skrócony właściwości w postaci `margin` i `padding`, za pomocą którego ustawia się wszystkie cztery właściwości za jednym razem:

```
margin: 0 10px 10px 20px;
padding: 10px 5px 5px 10px;
```

Wskazówka: Jeśli wartość właściwości CSS wynosi `0`, to nie trzeba podawać żadnej jednostki miary. Wystarczy na przykład napisać `margin: 0;` zamiast `margin: 0px;`.

Kolejność, w jakiej podane zostaną wszystkie cztery wartości, ma znaczenie i powinna być taka: *góra*, *prawa*, *dół*, *lewa*. Jeśli pomylimy kolejność, możemy mieć kłopoty. Aby ją zapamiętać, można sobie skojarzyć, że jest zgodna z ruchem wskazówek zegara i zaczyna się w samo południe (albo o północy).

Zastosowanie tej samej wartości do wszystkich czterech stron jest jeszcze prostsze — wystarczy podać tylko jedną wartość. Aby usunąć marginesy ze wszystkich znaczników `<h1>`, można napisać taki styl:

```
h1 {
    margin: 0;
}
```

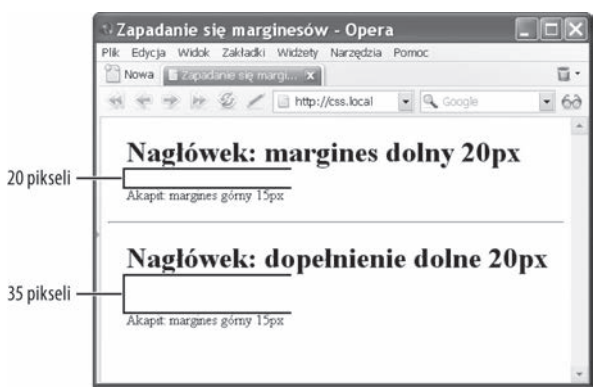
W podobny sposób tworzy się jednakowy odstęp pomiędzy treścią a obramowaniem ze wszystkich stron elementu:

```
padding: 10px;
```

Uwaga: Stosując takie same wartości dla marginesów górnego i dolnego oraz dwie takie same dla lewego i prawego, można użyć tylko dwóch wartości. Styl `margin 0 2em;` ustawia marginesy górny *i* dolny na `0`, a lewy *i* prawy na `2em`. Analogicznie jeśli górny i dolny margines (albo dopełnienie) różnią się, ale lewy i prawy są jednakowe, można użyć trzech wartości. Na przykład deklaracja `margin: 0 2em 1em` ustawia górny margines na `0`, lewy i prawy na `2em`, a dolny na `1em`.

Konflikty marginesów

W CSS dwa plus dwa nie zawsze równa się cztery. Gdy stykają się margines dolny elementu z marginesem górnym innego elementu, to próbując obliczyć, co się wtedy dzieje, trzeba by było zastosować jakieś nowe zasady matematyki. Zamiast dodać do siebie wartości marginesów, przeglądarka zastosuje tylko ten większy (rysunek 7.3, na górze). Załóżmy, że margines dolny listy nienumerowanej wynosi 30 pikseli, a akapitu znajdującego się pod nią 20 pikseli. Zamiast dodać te dwie wartości do siebie i utworzyć margines o szerokości 50 pikseli, przeglądarka zastosuje tylko ten większy — w tym przypadku 30 pikseli. Aby zapobiec temu zjawisku, można stosować tylko dopełnienie górne i dolne (rysunek 7.3, na dole).

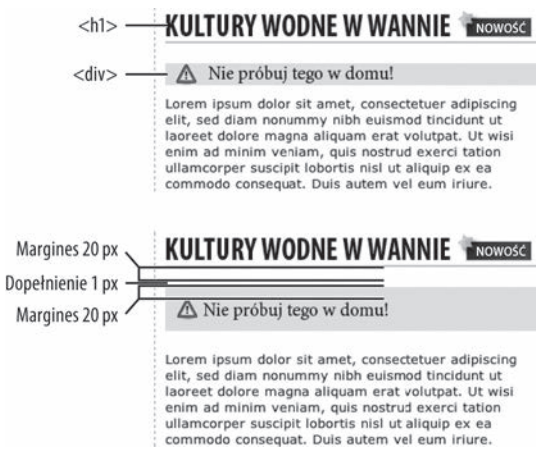


Rysunek 7.3. *Gdy stykają się dwa marginesy w pionie, to mniejszy z nich zapada się w tym większym. Mimo że górny nagłówek ma margines dolny o szerokości 20 pikseli, a akapit górny o szerokości 15 pikseli, przeglądarka utworzy tylko dwudziestopikselowy odstęp między nimi. Aby uzyskać 35 pikseli odstępu, w takim przypadku trzeba zastosować dopełnienie zamiast marginesów, jak pokazano na dolnym nagłówku. W tym przypadku nagłówek ma dwudziestopikselowe dopełnienie dolne. Tych 20 pikseli zostało dodanych do 15 pikseli marginesu górnego akapitu, co dało w sumie trzydziestopięciopikselowy odstęp*

Wszystko staje się jeszcze dziwniejsze, gdy jeden element znajduje się *wewnątrz* drugiego. Tego typu sytuacje mogą wywołać zachowania takie, jak drapanie się po głowie z zakłopotaniem. Załóżmy na przykład, że mamy na stronie ramkę

z ostrzeżeniem (przechowywanym np. w znaczniku). Dodajemy dwudziestopikselowy margines oddzielający tę ramkę od znajdującego się nad nią nagłówka i znajdującego się pod nią akapitu. Jak na razie wszystko jest w porządku.

Ale teraz wstawiamy do tej ramki nagłówek, który chcemy odsunąć od górnej i dolnej jej krawędzi na odległość 10 pikseli, więc ustawiamy marginesy o takich właśnie wartościach. Wydaje nam się, że dodaliśmy margines pomiędzy ramką a znajdującym się w niej nagłówkiem, ale jesteśmy w błędzie (rysunek 7.4). Zamiast tego margines pojawia się nad znacznikiem . W tym przypadku nie ma znaczenia rozmiar marginesu zastosowanego do nagłówka — i tak zawsze będzie znajdował się nad znacznikiem .



Rysunek 7.4. *Ginące piksele! Po lewej: jeśli dwa marginesy zetkną się w pionie, nawet gdy jeden element znajduje się wewnątrz drugiego, to ten mniejszy zapada się w większym. Po prawej: aby rozwiązać ten problem, można dodać niewielkie dopełnienie lub obramowanie wokół elementu nadrzędnego (w tym przypadku znacznika)*

Uwaga: W CSS zjawisko to określa się terminem scalania marginesów (ang. *collapsing margins*), co oznacza, że dwa marginesy zamieniają się w jeden.

Z problemem tym można sobie poradzić na dwa sposoby: można dodać niewielkie dopełnienie wokół znacznika lub obramowanie. Jako że obramowanie i dopełnienie znajdują się *pomiędzy* dwoma marginesami, zapobiegają ich stykaniu się, dzięki czemu nagłówek zyskuje nieco przestrzeni (rysunek 7.4, po prawej).

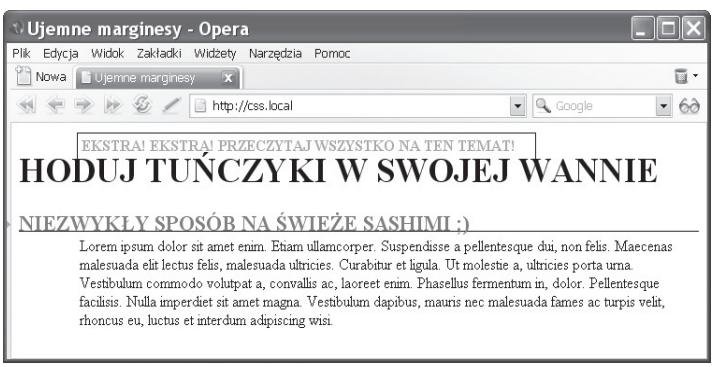
Uwaga: Marginesy stykające się w poziomie (lewy i prawy) oraz pomiędzy elementami pływającymi nie zapadają się, podobnie jak marginesy elementów pozycjonowanych względnie i bezwzględnie, o których będziemy mówić w rozdziale 15.

Likwidowanie odstępu za pomocą marginesów ujemnych

Większość wartości w CSS musi być dodatnia — bo jak miałby wyglądać tekst o rozmiarze pisma *–20 pikseli*? Dopełnienie również musi być wartością dodatnią. Ale jeśli chodzi o marginesy, to w CSS istnieje wiele przydatnych technik z wykorzystaniem ich wartości ujemnych.

Zamiast tworzyć odstęp między znacznikiem a elementami go otaczającymi, margines ujemny go *likwiduje*. Dzięki temu można sprawić, aby tekst akapitu nakładał się na nagłówek, wyszedł poza zawierający go element (pasek boczny lub inny znacznik `<div>` tworzący układ strony), a nawet zniknął za krawędzią okna przeglądarki. Z ujemnych marginesów naprawdę można mieć wiele pożytku.

Nawet jeśli marginesy pomiędzy nagłówkami zostaną ustawione na `0`, to i tak pomiędzy nimi zostanie niewielki odstęp (dzięki właściwości `line-height` opisanej w poprzednim rozdziale). W zasadzie jest to bardzo dobra rzecz, ponieważ tekst, w którym zdania stykają się ze sobą, czyta się bardzo trudno. Jednak przy zachowaniu odpowiedniego umiaru takie zmniejszenie odstępu między nagłówkami może przynieść interesujące rezultaty. Drugi nagłówek na rysunku 7.5 (zaczynający się od słów „Hoduj tuńczyki…") ma margines górny ustawiony na `-10px`. Powoduje to przesunięcie tego nagłówka do góry o 10 pikseli, dzięki czemu nachodzi on nieznacznie na przestrzeń zajmowaną przez nagłówek znajdujący się nad nim. Ponadto lewa i prawa krawędź nagłówka „Ekstra! Ekstra!" dotykają liter tego większego nagłówka.



Rysunek 7.5. *W tym przykładzie, aby górna krawędź ostatniego akapitu wyglądała jak dolna krawędź znajdującego się nad nim nagłówka, do akapitu trzeba dodać dopełnienie. Około 5 pikseli dopełnienia górnego odsuwa treść akapitu od krawędzi, a 4 em dopełnienia lewego tworzy wcięcie tekstu, pozwalając jednak krawędzi rozciągnąć się do końca w lewo*

Za pomocą ujemnych marginesów można symulować ujemne dopełnienie. W trzecim nagłówku na rysunku 7.5 (tym zaczynającym się od słów „Niezwykły sposób…") linia znajduje się bezpośrednio pod tekstem. W rzeczywistości jest to górna krawędź akapitu znajdującego się pod tym nagłówkiem (jak tworzyć obramowanie, dowiemy się w dalszej części rozdziału). Akapitowi temu ustawiono ujemny margines, co spowodowało jego przesunięcie do góry pod sam tekst nagłówka. Zwróć uwagę, że ogonek średnika wystaje nieco *pod* krawędzią. Jako że dopełnienie (przestrzeń między obramowaniem a treścią) nie może przyjmować wartości ujemnych, nie można przesunąć dolnej krawędzi do góry nad tekst czy cokolwiek innego. Ale ten sam efekt można uzyskać, nadając górną krawędź następnemu elementowi i przesuwając go do góry za pomocą ustawień marginesu.

Wskazówka: Ujemną wartość można w tym przypadku zastosować albo do dolnego marginesu nagłówka, albo do górnego marginesu akapitu. Każde z tych rozwiązań spowoduje taki sam efekt przesunięcia akapitu bliżej nagłówka.

Elementy śródliniowe, blokowe i inne

Mimo że przeglądarka wszystkie znaczniki traktuje jako pola, to nie wszystkie one są jednakowe. W CSS dostępne są dwa rodzaje pól: *blokowe* (ang. *block box*) i *śródliniowe* (ang. *inline box*), które odpowiadają dwóm typom znaczników: śródliniowym i blokowym.

Znacznik blokowy ma przed i za sobą odstęp. Znacznik `<p>` na przykład tworzy blok, który jest oddzielony wolną przestrzenią od znaczników znajdujących się nad i pod nim. Innymi „przedstawicielami" elementów blokowych są znaczniki `<div>`, tabele i listy.

Znaczniki śródliniowe nie tworzą odstępu nad i pod sobą. Występują w tej samej linii co pozostała treść i znaczniki obok nich. Do tego typu należy znacznik `<strong>`. Słowo sformatowane za pomocą tego znacznika przebywa w miłym towarzystwie reszty tekstu, a nawet innych znaczników śródliniowych, jak `<em>`. W rzeczywistości gdyby wyróżnione słowo ze środka akapitu nagle znalazło się samo w wierszu na stronie, to wyglądałoby to odrobinę dziwnie. Inne znaczniki śródliniowe to: `<img>` dodający obrazy na stronę, `<a>` tworzący odnośniki i różne znaczniki służące do tworzenia pól formularza.

W większości przypadków arkusze stylów zachowują się tak samo wobec elementów blokowych, jak i śródliniowych. Można zmieniać krój i kolor pisma, kolor tła i ramek. Jeśli jednak chodzi o marginesy i dopełnienie, to elementy śródliniowe są traktowane przez przeglądarki inaczej niż blokowe. Podczas gdy do elementu śródliniowego można dodać lewy lub prawy margines albo lewe lub prawe dopełnienie, to nie można zwiększyć jego wysokości za pomocą górnego marginesu lub dopełnienia. W górnym akapicie na rysunku 7.6 element śródliniowy ma obramowanie, zmieniony kolor tła i dwudziestopikselowy margines z wszystkich czterech stron. Przeglądarka jednak odstęp dodała tylko po jego lewej i prawej stronie.

Uwaga: Jedynym wyjątkiem, kiedy elementy śródliniowe nie robią się wyższe po zastosowaniu dopełnienia lub marginesu, jest znacznik `<img />` (mimo że jest on znacznikiem śródliniowym). Przeglądarki poprawnie rozszerzają wysokość pola obrazu, aby pomieścić margines i dopełnienie.

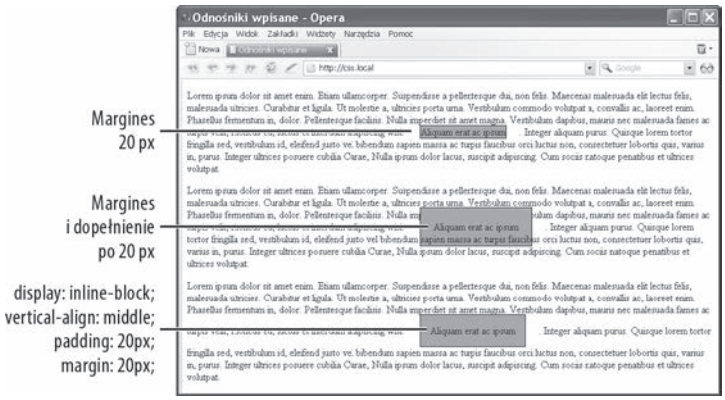
Czasami możemy zechcieć, aby element śródliniowy zachowywał się jak blokowy i odwrotnie. Elementy list wypunktowanych są oddzielnymi elementami blokowymi — ustawione są jeden nad drugim. Ale co zrobić, jeśli chcemy to zmienić w taki sposób, aby elementy listy znajdowały się obok siebie w jednej linii (jak w pasku nawigacyjnym, który zobaczymy w rozdziale 9.)? Możemy też zechcieć potraktować element śródliniowy jako blokowy, na przykład jeśli chcemy, aby obrazek w akapicie znajdował się sam w wierszu i miał górny i dolny margines.

Na szczęście w CSS jest właściwość, która pozwala to zrobić: `display`. Za jej pomocą można sprawić, aby element blokowy zachowywał się jak element śródliniowy:

```
display: inline;
```

lub aby element śródliniowy zachowywał się jak blokowy:

```
display: block;
```



Rysunek 7.6. *Stosowanie górnego marginesu i dopełnienia do elementów śródliniowych nie wywołuje żadnego efektu, przez co można uzyskać nieoczekiwane rezultaty. W środkowym akapicie tło i obramowanie odnośnika nakładają się na tekst znajdujący się nad i pod nim. Tło zasłania tekst w wierszu znajdującym się nad stylizowanym elementem, ale nie pod następnym wierszem, ponieważ przeglądarka każdą linię traktuje w taki sposób, jakby była ona ułożona na wierzchu tej, która jest pod nią. W normalnych warunkach nie sprawia to żadnego problemu, ponieważ wiersze tekstu z reguły nie nachodzą na siebie. Aby można było ustawiać górny i dolny margines oraz dopełnienie dla elementów śródliniowych, można zastosować do nich deklarację display: inline-block (na dole). Dzięki niej element pozostaje śródliniowy, ale jest traktowany jak blokowy, tzn. można mu ustawiać marginesy, dopełnienie, obramowanie, szerokość i wysokość. Własność ta działa nawet w Internet Explorerze 7, ale tylko dla normalnych elementów śródliniowych, takich jak , i . Należy też dodać deklarację vertical-align: middle, aby przeglądarka IE 7 wyświetlała element śródliniowo-blokowy tak samo jak inne przeglądarki*

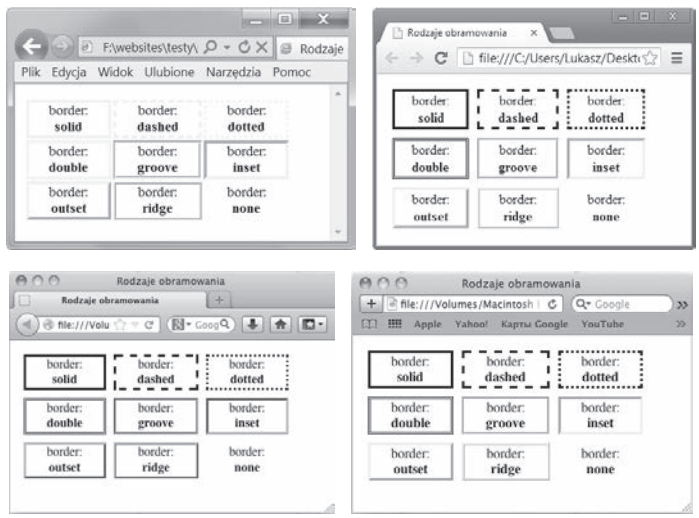
Uwaga: Właściwość `display` może przyjmować mnóstwo wartości, z których część nie jest jeszcze obsługiwana przez wszystkie przeglądarki. Wartość `inline-block` rozpoznają wszystkie aktualnie używane przeglądarki (rysunek 7.6). Jest jednak wartość `none`, którą obsługują prawie wszystkie przeglądarki i która ma bardzo wiele zastosowań. Wykonuje ona jedno proste zadanie — ukrywa cały element, tak że nie widać go w oknie przeglądarki. Przy użyciu języka JavaScript można sprawić, że ukryty w ten sposób element pojawi się na stronie za sprawą zmiany wartości `display` na `inline` lub `block`. W CSS można też sprawić, aby ukryty element nagle pojawił się na stronie. W dodatku A pokazany jest stosowny przykład.

Obramowanie

Obramowanie to linia biegnąca wokół elementu. Jak widać na rysunku 7.1, znajduje się ona pomiędzy dopełnieniem a marginesem. Obramowanie otaczające obraz może stanowić jego obramowanie lub wyznaczać granice banera albo innego elementu strony. Obramowanie nie musi jednak otaczać elementów z wszystkich stron. Można łatwo otoczyć obramowaniem element z wszystkich stron lub w równie prosty sposób zastosować je tylko do dowolnie wybranych krawędzi. Dzięki tej elastyczności można tworzyć takie efekty wizualne, które niekoniecznie mają coś wspólnego z obramowaniem. Jeśli na przykład po lewej stronie elementu dodamy obramowanie o szerokości 1 em, to będzie ono wyglądało jak kwadratowy punktor. Pojedyncze obramowanie pod akapitem może pełnić rolę znacznika `<hr>` (linia pozioma), wizualnie oddzielając dwie części strony.

Obramowanie ma trzy właściwości, które można kontrolować: `color`, `width` i `style`. Kolor można wyrazić w postaci liczby szesnastkowej, słowa kluczowego lub wartości RGB albo RGBA (tak samo jak w przypadku tekstu). Do definiowania grubości obramowania można stosować wszystkie jednostki miary CSS poza procentami oraz słowa kluczowe: `thin` (cienka), `medium` (średnia) i `thick` (gruba). Najczęściej stosowaną i najłatwiejszą do zrozumienia metodą jest stosowanie pikseli.

Za pomocą stylów można też kontrolować rodzaj rysowanej linii. Rodzajów jest wiele, a wygląd niektórych z nich dość znacznie różni się w różnych przeglądarkach, jak widać na rysunku 7.7. Styl linii określa się za pomocą słów kluczowych. Na przykład słowo `solid` oznacza linię ciągłą, a `dashed` przerywaną złożoną z krótkich kresek. W CSS dostępne są następujące słowa kluczowe określające styl ramek: `solid`, `dotted`, `dashed`, `double`, `groove`, `ridge`, `inset`, `outset`, `none` oraz `hidden`. (Słowa kluczowe `none` i `hidden` działają tak samo — całkiem usuwają obramowanie. Słowo `none` jest przydatne przy wyłączaniu jednej krawędzi.



Rysunek 7.7. *Wygląd różnych stylów ramek może się znacznie różnić w różnych przeglądarkach. Przeglądarki widoczne na tym rysunku to Internet Explorer 9 (na górze po lewej), Chrome dla Windowsa (na górze po prawej), Firefox dla systemu Mac (na dole po lewej) i Safari dla Maca (na dole po prawej)*

Skrócony zapis właściwości obramowania

Patrząc na listę różnych właściwości obramowania dostępnych w CSS, można odnieść wrażenie, że są one bardzo skomplikowane. W końcu jest aż dwadzieścia różnych właściwości ramek, o których będziemy mówić w tym podrozdziale, oraz kilka takich, które odnoszą się do tabel. Wszystkie te właściwości to jednak tylko różne kombinacje trzech podstawowych właściwości: koloru, szerokości i stylu dla każdej z czterech krawędzi. Podstawową i najprostszą właściwością jest `border`, która po prostu tworzy pełne obramowanie:

```
border: 4px solid #FF0000;
```

Powyższy styl tworzy ciągłe obramowanie o szerokości 4 pikseli i czerwonym kolorze. Za jego pomocą można otoczyć obramowaniem obrazek, pasek nawigacyjny lub jakiś inny element, gdy chcemy, aby występował jako samodzielny blok.

Uwaga: Kolejność wpisywania właściwości nie ma tu znaczenia — `border: 4px solid #FF0000` jest tak samo dobre jak `border: #FF0000 solid 4px`.

Formatowanie poszczególnych krawędzi

Każdą krawędź elementu można formatować indywidualnie, stosując odpowiednią właściwość: `border-top`, `border-bottom`, `border-left` lub `border-right`. Właściwości te funkcjonują dokładnie tak samo jak właściwość `border`, ale odnoszą się tylko do jednej krawędzi. Poniższa właściwość rysuje dwupikselową czerwoną kreskowaną linię pod elementem, do którego zostanie zastosowana:

```
border-bottom: 2px dashed red;
```

Właściwość `border` można stosować w połączeniu z właściwościami przeznaczonymi dla konkretnych krawędzi. Można w ten sposób określić podstawowy styl obramowania całego elementu, a potem odpowiednio dostosować tylko jedną wybraną krawędź. Załóżmy, że chcemy, aby krawędzie górna, lewa i prawa były takie same, ale dolna wyglądała nieco inaczej. Efekt ten można uzyskać przy użyciu czterech poniższych wierszy kodu CSS:

```
border-top: 2px solid black;
border-left: 2px solid black;
border-right: 2px solid black;
border-bottom: 4px dashed #333;
```

lub tylko dwóch:

```
border: 2px solid black;
border-bottom: 4px dashed #333;
```

Pierwszy wiersz kodu definiuje wygląd wszystkich czterech krawędzi, a drugi zmienia prezentację tylko dolnej. Korzyści, jakie z tego płyną, to nie tylko mniejsza ilość kodu do wpisania, ale też łatwiejsze wprowadzanie zmian w stylach. Jeśli chcemy zmienić kolor lewej, górnej lub prawej krawędzi na czerwony, to musimy tylko zmodyfikować jeden wiersz zamiast trzech:

```
border: 2px solid red;
border-bottom: 4px dashed #333;
```

Przy zastosowaniu tej skrótowej metody — polegającej na zdefiniowaniu wyglądu całego obramowania za pomocą właściwości `border`, a następnie zmianie wyglądu tylko jednej krawędzi za pomocą odpowiedniej właściwości, na przykład `border-left` — bardzo ważną rolę odgrywa kolejność wpisywania właściwości. Właściwość bardziej ogólna musi wystąpić przed tą bardziej specyficzną, jak poniżej:

```
border: 2px solid black;
border-bottom: 4px dashed #333;
```

Jako że właściwość `border-bottom` jest druga, to przesłania ona ustawienie właściwości `border`. Gdyby właściwość `border-bottom` znalazła się przed `border`, to zostałaby przez nią przesłonięta i wszystkie cztery krawędzie byłyby identyczne. Ostatnia z właściwości przesłania wszystkie te, które z nią kolidują. Jest to przykład kaskadowości CSS, o której pisałem w rozdziale 5.

Tej samej skrótowej metody można użyć do wyłączenia jednej krawędzi za pomocą słowa kluczowego `none`. Załóżmy, że chcemy otoczyć element obramowaniem z trzech

stron (na górze, z lewej i na dole), a czwartą (prawą) pozostawić bez obramowania. Możemy to osiągnąć przy użyciu tylko dwóch wierszy kodu:

```
border: 2px inset #FFCC33;
border-right: none;
```

Tak duża liczba właściwości ramek jest podyktowana możliwością ich precyzyjnego dostosowywania. Pozostałe piętnaście z nich pozwala na definiowanie indywidualnych kolorów, stylów i szerokości całego obramowania, jak i poszczególnych jego krawędzi. Właściwość `border: 2px double #FFCC33` można na przykład zapisać tak:

```
border-width: 2px;
border-style: double;
border-color: #FFCC33;
```

Jako że metoda ta wymaga użycia aż trzech wierszy kodu zamiast jednego, będziemy starali się jej unikać. Każda krawędź ma jednak własny zestaw trzech właściwości, które pozwalają na zdefiniowanie tylko jednej cechy każdej z czterech krawędzi obramowania. Prawa krawędź ma następujące trzy właściwości: `border-right-width`, `border-right-style` i `border-right-color`. Pozostałe krawędzie mają podobne właściwości: `border-left-width`, `border-left-style` itd.

Grubość tylko jednej krawędzi można zmienić tak: `border-right-width: 4px;`. Jedną z zalet takiego podejścia jest to, że jeśli później postanowimy zmienić styl obramowania na linię ciągłą, będziemy musieli zmodyfikować tylko generyczną właściwość `border`, zamieniając słowo kluczowe `dashed` na `solid`.

Dodatkowo można też zdefiniować wartość dla każdej krawędzi obramowania przy użyciu własności `border-width`, `border-style` oraz `border-color`. Na przykład deklaracja `border-width: 10px 5px 15px 13px` każdej z krawędzi (górnej, prawej, dolnej i lewej) ustawia inną grubość.

Wyobraźmy sobie, że chcemy, aby wszystkie cztery krawędzie elementu były przerywane i miały grubość 2 pikseli, ale każda z nich musi mieć inny kolor (np. na stronie dla dzieci). Oto szybki sposób na osiągnięcie tego efektu:

```
border: 2px dashed green;
border-color: green yellow red blue;
```

Powyższe reguły tworzą przerywane obramowanie o grubości 2 pikseli, nadając górnej krawędzi kolor zielony, prawej żółty, dolnej czerwony, a lewej niebieski.

Uwaga: Stosując obramowanie, z reguły stosuje się też dopełnienie. Tworzy ono odstęp pomiędzy obramowaniem a treścią, taką jak tekst, obrazy lub inne znaczniki. Bez dopełnienia obramowanie zazwyczaj znajduje się zbyt blisko treści.

Kolorowanie tła

Nadanie koloru tła całej stronie, jednemu nagłówkowi lub jakiemukolwiek innemu elementowi na stronie to pestka. Do tego celu należy wykorzystać właściwość `background-color` oraz podać kolor na jeden ze sposobów opisanych w rozdziale 6. Jeśli chcemy, to możemy za pomocą poniższego kodu zmienić kolor tła całej strony na jaskrawozielony:

```
body { background-color: rgb(109,218,63); }
```

Alternatywnie można utworzyć klasę o nazwie, na przykład, `.review`, zawierającą właściwość `background-color`, a następnie zastosować ją do znacznika HTML `<body>`: `<body class="review">`.

Uwaga: W tle strony można też umieścić dowolny obraz, a nawet sterować jego położeniem na wiele różnych sposobów. Więcej na ten temat piszę w następnym rozdziale. Dodatkowo w tle każdego elementu można zdefiniować gradient.

Zmiana kolorów tła umożliwia osiągnięcie wielu różnych efektów wizualnych. Można na przykład stworzyć krzykliwy nagłówek, ustawiając kolor jego tła na jakiś ciemny odcień, a tekstu na jasny. Zmiana koloru tła doskonale nadaje się również do oddzielenia części strony, na przykład paska nawigacyjnego, banera czy paska bocznego, od jej reszty.

Nie zapomnij też o opisanym w rozdziale 6. formacie RGBA. Dzięki niemu można zdefiniować częściowo przezroczyste tło, przez które będą przeświwitywać znajdujące się pod spodem kolory, tekstury i obrazy. Można na przykład dla całej strony ustawić jasnobrązowe tło. Jeśli później chcemy utworzyć element `<div>` w trochę jaśniejszym kolorze, zamiast definiować nowy kolor w tym elemencie, można ustawić biały i dostosować jego poziom przezroczystości, aby uzyskać odpowiedni stopień rozjaśnienia znajdującego się pod spodem brązu:

```
body {
    background-color: rgb(247,226,155);
}
.special-div {
    background-color: rgba(255,255,255.75);
}
```

Wskazówka: Stosując kolory tła i obramowania, należy pamiętać o jednej rzeczy: jeśli linia obramowania jest przerywana (zobacz rysunek 7.7), to w przerwach będzie widoczne tło. Innymi słowy, przeglądarki malują tło *pod* obramowaniem.

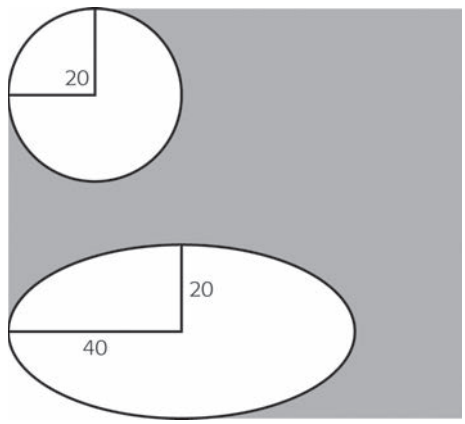
Zaokrąglanie rogów

Wcześniej napisałem, że dla przeglądarek każdy element jest prostokątem, o czym można się łatwo przekonać, definiując dowolnemu elementowi obramowanie. Można jednak te kanty ładnie zaokrąglić, aby wyglądały tak, jak na przykład na rysunku 7.8. W CSS3 wprowadzono nową własność o nazwie `border-radius`, służącą do definiowania zaokrąglenia dowolnej liczby rogów wybranego elementu. W najprostszej postaci własność ta przyjmuje jedną wartość, która jest stosowana do wszystkich czterech rogów elementu:

```
.specialBox {
    background-color: red;
    border-radius: 20px;
}
```

Przeglądarka rysuje okrąg wokół każdego rogu elementu. Wartość własności `border-radius` określa odległość od środka tego okręgu do jego obwodu — jego promień — jak pokazano na rysunku 7.9. Najczęściej używanymi jednostkami są piksele i `em`, ale można też używać procentów (chociaż działają trochę inaczej, niż można by było się spodziewać — zobacz podpis do rysunku 7.9).

Rysunek 7.8. *W CSS3 można zaokrąglić rogi dowolnego elementu. Aby efekt był widoczny, element musi mieć zdefiniowane kolorowe tło lub obramowanie*



Rysunek 7.9. *Zaokrąglenia rogów mogą być okrągłe (na górze) lub eliptyczne, w zależności od tego, czy własności border-radius zostanie przekazana jedna wartość, np. 20px, czy dwie oddzielone ukośnikiem, np. 40px/20px. Zastosowanie wartości procentowej zazwyczaj powoduje powstanie zaokrąglenia eliptycznego, ponieważ przeglądarka oblicza długość poziomego promienia w odniesieniu do szerokości elementu, a pionowego — w odniesieniu do jego wysokości. Dlatego przykładowo dla deklaracji border-radius: 20%, jeśli nie zostanie zastosowana do kwadratowego elementu, zostanie utworzone zaokrąglenie eliptyczne, takie jak np. border-radius: 20px/40px*

Jeśli podana jest tylko jedna wartość, każdy róg jest zaokrąglany w taki sam sposób. Przykładowo element znajdujący się w lewym górnym rogu na rysunku 7.8 został zdefiniowany przy użyciu poniższej deklaracji:

```
border-radius: 30px;
```

Ale nie wszystkie rogi muszą być takie same. Zaokrąglenie każdego z nich można zdefiniować oddzielnie, wpisując cztery wartości. Na przykład do elementu znajdującego się w prawym górnym rogu na rysunku 7.8 zastosowano poniższą deklarację:

```
border-radius: 0 30px 10px 5px;
```

Pierwsza wartość odnosi się do lewego górnego rogu, a następne do kolejnych rogów zgodnie z ruchem wskazówek zegara. Innymi słowy, pierwsza wartość (na rysunku 7.8 wynosząca `0`) dotyczy lewego górnego rogu, druga wartość (`30px`) dotyczy prawego górnego rogu, trzecia wartość (`10px`) dotyczy prawego dolnego rogu, a czwarta (`5px`) — lewego dolnego. Można też wpisać tylko dwie wartości i wówczas pierwsza odnosi się do lewego górnego i dolnego prawego rogu, a druga — do prawego górnego i lewego dolnego.

Oprócz idealnie okrągłych zaokrągleń można też definiować zaokrąglenia eliptyczne, takie jak w dwóch elementach na dole na rysunku 7.8. Aby zdefiniować eliptyczne zaokrąglenie, trzeba podać dwie wartości. Pierwsza z nich określa długość promienia poziomego, a druga — pionowego. Przykładowo rogi elementu znajdującego się w lewym dolnym rogu na rysunku 7.8 zdefiniowano przy użyciu poniższej deklaracji:

```
border-radius: 40px/20px;
```

ABY ZYSKAĆ NA CZASIE

Przedrostki producentów

CSS to ciągle zmieniający się zestaw własności. Nawet w tej chwili bystrzaki z W3C (ang. *World Wide Web Consortium*) pracują nad nowymi własnościami, a inni inteligenci próbują zaimplementować ich obsługę w przeglądarkach internetowych. Czasami nawet twórcy przeglądarek sami wymyślają jakieś własności, które według nich mogą być przydatne, i dodają je do swoich produktów. Zdarza się też, że niektóre własności opracowywane przez W3C wydają się właścicielom przeglądarek mało potrzebne, co sprawia, że długo czekają one na implementację.

W czasie gdy standard CSS jest dopiero w fazie powstawania, twórcy przeglądarek działają ostrożnie, ponieważ nie chcą zainwestować zbyt dużo czasu w implementację własności, które mogą się jeszcze zmienić. Także gdy producent przeglądarki eksperymentuje z wymyśloną przez siebie własnością, nie udaje, że jest to powszechnie akceptowany standard. Tego typu własności, tzn. eksperymentalne lub zaimplementowane na próbę, są poprzedzane specjalnym przedrostkiem. Każdy z najważniejszych producentów przeglądarek internetowych ma swój własny przedrostek:

- Przedrostek `-webkit-` jest używany w przeglądarkach Chrome, Safari i innych opartych na algorytmie WebKit.
- Przedrostek `-moz-` jest używany w przeglądarce Mozilla Firefox.
- Przedrostek `-o-` oznacza własność Opery.
- Przedrostek `-ms-` należy do przeglądarki Microsoft Internet Explorer.

Gdy specyfikacja własności nie jest jeszcze ukończona, w przeglądarkach może pojawić się jej próbna implementacja z przedrostkiem. Na przykład gdy przedstawiono wstępną dokumentację własności `border-radius`, w Firefoksie zaimplementowano ją pod nazwą `-moz-border-radius`, a w Safari jako `-webkit-border-radius`.

Gdy jakaś własność musi być zapisywana z przedrostkiem, to zazwyczaj trzeba napisać ją w kilku wersjach: po jednej z każdym przedrostkiem i na końcu bez przedrostka:

```
-moz-box-sizing: border-box;
-webkit-box-sizing: border-box;
box-sizing: border-box;
```

Zazwyczaj gdy W3C zaadaptuje własność i poda wystarczającą liczbę szczegółów technicznych dotyczących sposobu jej implementacji, wersja z przedrostkiem zostaje zamieniona na bez przedrostka. Na przykład aktualnie wszystkie najważniejsze przeglądarki obsługują własność `border-radius` bez przedrostka. Jednak w dalszej części książki poznasz sporo własności CSS3, które wciąż są „szlifowane" i może być konieczne zastosowanie przedrostka, aby ich używać. Tam, gdzie była taka potrzeba, podałem stosowne informacje i wskazówki na temat sposobu użycia tych własności.

Wartość `40px` określa długość promienia poziomego, a `20px` — pionowego. Ukośnik stanowi znak dla przeglądarki, że chcemy utworzyć zaokrąglenie eliptyczne. Można też każdemu narożnikowi zdefiniować inne zaokrąglenie. W tym celu należy podać cztery pary wartości:

```
border-radius: 40px/20px 10px/30px 20px/40px 10px/20px;
```

Można nawet mieszać definicje okrągłych i eliptycznych rogów w jednej deklaracji:

```
border-radius: 10px 10px/30px 20px/40px 10px;
```

Jeśli lubisz dużo pisać, możesz również zdefiniować każdy narożnik przy użyciu osobnej własności. Na przykład poniższą deklarację:

```
border-radius: 1em 2em 1.5em .75em;
```

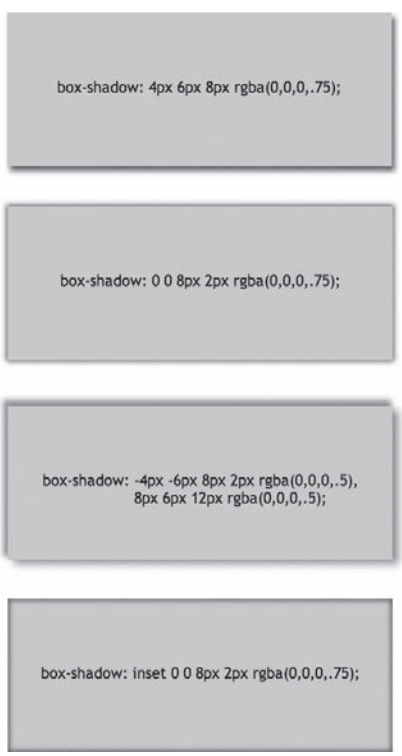
można zamienić na poniższe cztery:

```
border-top-left-radius: 1em;
border-top-right-radius: 2em;
border-bottom-right-radius: 1.5em;
border-bottom-left-radius: .75em;
```

Uwaga: Przeglądarka Internet Explorer do wersji 8 włącznie nie obsługuje własności `border-radius`, a więc wyświetli kanciaste rogi zamiast zaokrąglonych. Ponadto przeglądarki Safari 3.2 dla systemu iOS i Android Browser 2.1 wymagają przedrostka `-webkit-`, tzn. należy na przykład pisać `-webkit-border-radius: 20px`. Szczegółowe informacje na ten temat znajdują się w ramce "„Przedrostki producentów".

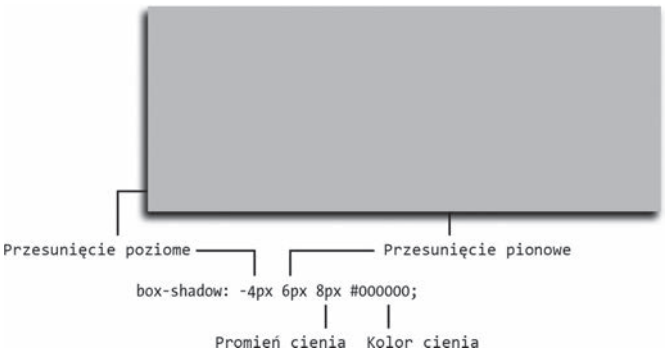
Cienie elementów

W rozdziale 6. dowiedziałeś się, jak dodawać cień do tekstu, aby go wyróżnić spośród innych tekstów na stronie. Jednak w CSS3 dostępna jest też własność `box-shadow` służąca do definiowania cieni dla pól elementów, dzięki czemu można sprawić, że elementy wyglądają, jakby unosiły się nad stroną (rysunek 7.10). Własność ta ma trochę więcej opcji niż wcześniej opisana `text-shadow`. Za jej pomocą można na przykład zdefiniować cień wewnątrz pola elementu, jak na dole na rysunku 7.10.



Rysunek 7.10. *Własność CSS3 box-shadow służy do tworzenia cieni elementów, dzięki którym wyglądają one tak, jakby unosiły się nad stroną. Działa we wszystkich nowych przeglądarkach, włącznie z Internet Explorerem od wersji 9, chociaż cienie w tym programie są znacznie cieńsze niż w pozostałych przeglądarkach. Internet Explorer od wersji 8 w dół ignoruje własność box-shadow*

Podstawowa składnia własności `box-shadow` jest przedstawiona na rysunku 7.11. Pierwsza wartość określa przesunięcie poziome, tzn. przesuwa cień na określoną odległość w prawo lub w lewo względem elementu. Dodatnia liczba powoduje przesunięcie w prawo (jak na górze na rysunku 7.10), a ujemna — w lewo.



Rysunek 7.11. *Najprostszy cień może mieć określony kolor i być ustawiony po lewej (ujemna wartość, jak na tym rysunku) lub prawej (dodatnia wartość) stronie elementu oraz nad (ujemna wartość) lub pod (dodatnia wartość) nim. Dodatkowo można zdefiniować rozmazanie cienia*

Druga wartość określa przesunięcie pionowe, tzn. przesuwa cień na określoną odległość do góry lub w dół względem elementu. Dodatnia liczba powoduje wysunięcie cienia pod dolną krawędzią elementu (jak na górze na rysunku 7.10), a ujemna — nad górną.

Uwaga: Do definiowania cieni elementów należy używać jednostek `em` lub pikseli. Procenty są niedozwolone.

Trzecia wartość to promień cienia i określa ona jego stopień rozmazania oraz szerokość. Zero oznacza brak rozmazania, dzięki czemu krawędzie cienia są ostre. Duża wartość powoduje utworzenie większego rozmazania i grubszego cienia. Ostatnia wartość określa kolor. Można używać dowolnego formatu dostępnego w CSS, ale najlepiej wyglądają kolory w formacie RGBA, ponieważ można ustawiać im stopień przezroczystości, dzięki czemu bardziej przypominają prawdziwy cień.

Własność `box-shadow` przyjmuje jeszcze dwie opcjonalne wartości: `inset` i rozmiar cienia. Słowo kluczowe `inset` sprawia, że cień jest rysowany wewnątrz pola elementu (jak na dole na rysunku 7.10). Słowo to wpisuje się na pierwszym miejscu listy wartości:

```
box-shadow: inset 4px 4px 8px rgba(0,0,0,.75);
```

Na czwartym miejscu (między promieniem a kolorem) można dodatkowo zdefiniować rozmiar cienia, tzn. jeśli wpiszemy `10px`, to cień zostanie rozciągnięty o 10 pikseli w każdym kierunku (co w efekcie sprawi, że będzie o 20 pikseli szerszy i 20 pikseli wyższy). Wartość ta decyduje też o zastosowaniu promienia rozmazania, tzn. rozmazanie rozpoczyna się dopiero po określeniu rozmiaru. Jest to szczególnie przydatne, gdy chcemy otoczyć cieniem cały element — taki efekt w programach graficznych zazwyczaj nazywa się *poświatą*.

Uwaga: Dla przeglądarek Android i starszych wersji Safari dla iPhone'a wymagana jest własność `box-shadow` z przedrostkiem (szczegółowe informacje na temat przedrostków własności znajdują się w ramce „Przedrostki producentów"). Krótko mówiąc, jeśli chcesz, aby cienie elementów były widoczne w tych i nowszych przeglądarkach, musisz napisać dwie deklaracje:

```
-webkit-box-shadow: 2px 2px 10px #000000;
box-shadow: 2px 2px 10px #000000;
```

Na przykład drugi element od góry na rysunku 7.10 ma przesunięcie poziome i pionowe ustawione na `0`. Promień jest ustawiony na `8px`, a rozmiar na `2px`. Wartość rozmiaru wypycha cień na odległość dwóch pikseli poza wszystkie cztery krawędzie elementu, a ośmiopikselowy promień rozmazania rozszerza cień jeszcze bardziej. Określając rozmiar cienia, można nawet zdefiniować dodatkowe kolorowe obramowanie elementu:

```
border: 10px solid rgb(100,255,30);
box-shadow: 0 0 0 10px rgb(0,33,255);
```

Element może mieć zdefiniowanych kilka cieni (rysunek 7.10 — drugi element od dołu). Poszczególne listy wartości należy w takim przypadku oddzielać przecinkami:

```
box-shadow: 10px 5px 8px #FF00FF,
            -5px -10px 20px 5px rgb(0,33,255);
```

Liczba cieni jest ograniczona tylko rozsądkiem projektanta.

Uwaga: Niestety, przeglądarki bardzo różnie rysują cienie elementów. Na przykład IE 9 tworzy znacznie cieńszy cień niż pozostałe przeglądarki. Dobre wizualne porównanie cieni w różnych programach znajduje się na stronie *http://thany.nl/apps/boxshadows/*.

Określanie wysokości i szerokości

Dwie następne właściwości, które wchodzą w skład modelu polowego CSS, pozwalają określać wymiary obiektów na stronie, takich jak tabele, kolumny, banery czy paski boczne. Właściwości `height` i `width` służą do określania wysokości i szerokości elementów. Będziemy je często wykorzystywać do tworzenia układów w CSS, które opisuję w części trzeciej tej książki, chociaż znajdują też zastosowanie w bardziej podstawowych zadaniach, takich jak zmiana szerokości tabeli, tworzenie prostego paska bocznego lub tworzenie galerii miniaturek obrazków.

Dodawanie tych właściwości do stylu jest bardzo łatwe. Wystarczy wpisać nazwę właściwości, a po niej wartość z jedną z opisywanych do tej pory jednostek miary. Na przykład:

```
width: 300px;
width: 30%;
height: 20em;
```

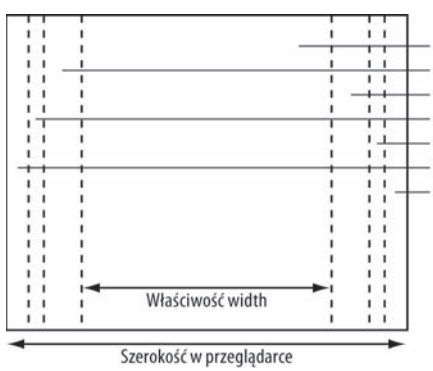
Piksele to… piksele. Są łatwe do opanowania i nie sprawiają problemów w zastosowaniu. Tworzą też dokładne wymiary, które się nie zmienią. Jednostka *em* odpowiada rozmiarowi tekstu w stylizowanym elemencie. Załóżmy, że ustawiliśmy rozmiar pisma na 24 piksele, jeśli więc szerokość ustawimy na 2 em, to powinna wynosić 2×24 px, czyli 48 pikseli. Jeśli w danym stylu nie określimy rozmiaru tekstu, to rozmiar jednostki em będzie zależny od odziedziczonego rozmiaru tekstu.

Uwaga: W CSS3 wprowadzono nową jednostkę o nazwie `rem`, której wartość jest zależna od rozmiaru pisma elementu `html`. Jednostka ta działa we wszystkich najważniejszych przeglądarkach oprócz Internet Explorera do wersji 8 włącznie. Więcej informacji na jej temat znajduje się w rozdziale 6.

W przypadku wartości procentowych szerokość jest równa odpowiedniej części szerokości elementu rodzica stylizowanego elementu. Jeśli szerokość nagłówka ustawimy na 75 procent i nagłówek ten nie będzie znajdował się w żadnym innym elemencie o ustalonej szerokości, to jego szerokość będzie wynosiła 75 procent szerokości okna przeglądarki. Jeśli użytkownik zmieni rozmiar okna, to szerokość nagłówka również ulegnie zmianie. Jeśli jednak nagłówek znajdowałby się w znaczniku `<div>` (np. reprezentującym kolumnę) o szerokości 200 pikseli, to jego szerokość wynosiłaby 150 pikseli. Wartości procentowe właściwości `height` funkcjonują tak samo, ale opierają się na wysokości elementu nadrzędnego, a nie na szerokości.

Obliczanie rzeczywistych wymiarów pól

Mimo że właściwości `width` i `height` są bardzo łatwe w użyciu, to jest kilka niuansów, które już niejednego przyprawiły o ból głowy. Po pierwsze pomiędzy zdefiniowaną szerokością i wysokością elementu a ilością miejsca, która w rzeczywistości zostanie mu przydzielona przez przeglądarkę, jest różnica. Właściwości te ustawiają wymiary *obszaru treści* elementu — miejsca, w którym znajdują się obrazy, tekst i inne zagnieżdżone znaczniki (aby sobie przypomnieć, gdzie znajduje się obszar treści elementu w modelu blokowym, spójrz na rysunek 7.1). Rzeczywista szerokość elementu — to znaczy ilość miejsca na ekranie przydzielona mu przez przeglądarkę — to suma szerokości, marginesów, obramowania, dopełnienia i właściwości `width`, jak pokazano na rysunku 7.12.



Szerokość	100 px
Dopełnienie lewe	15 px
Dopełnienie prawe	15 px
Krawędź lewa	5 px
Krawędź prawa	5 px
Margines lewy	10 px
Margines prawy	10 px
Szerokość całkowita	**160 px**

Rysunek 7.12. *Rzeczywistą szerokość elementu na stronie oblicza się, dodając do siebie szerokości jego marginesów, obramowania, dopełnienia i wartości właściwości width. Wysokość jest obliczana w taki sam sposób — jest to suma wartości właściwości height, marginesów górnego i dolnego, górnej i dolnej krawędzi oraz górnego i dolnego dopełnienia*

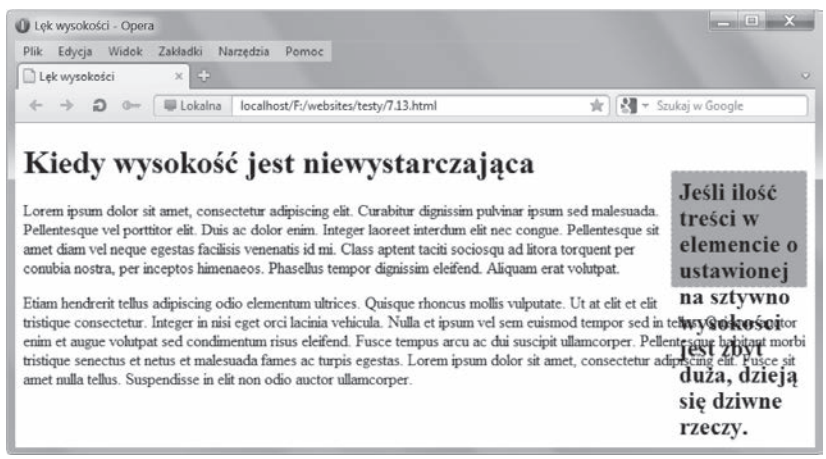
Wyobraźmy sobie, że ustawiliśmy te właściwości:

```
margin: 10px;
padding: 15px;
border-width: 5px;
width: 100px;
```

Ustawiając właściwość `width`, wiemy tylko, ile miejsca zostało przeznaczone dla samej treści — tekstu i obrazów wypełniających przestrzeń — bez względu na inne właściwości, które również mogły zostać ustawione. Nie trzeba wykonywać żadnych obliczeń, ponieważ wartość właściwości `width` określa ilość miejsca dla treści (w powyższym przykładzie 100 pikseli). Oczywiście aby określić, ile miejsca w rzeczywistości zajmie na stronie dany element, *trzeba* wykonać pewne obliczenia.

W powyższym przykładzie przeglądarka przydzieli elementowi obszar o szerokości 160 pikseli: 20 dla lewego i prawego marginesu, 10 dla krawędzi lewej i prawej, 30 dla lewego i prawego dopełnienia oraz 100 dla szerokości.

Podstawowa zasada dotycząca określania wysokości elementów brzmi: **nie rób tego!** Początkujący projektanci stron często definiują wysokość wszystkich elementów, aby mieć nad wszystkim kontrolę. Jeśli jednak nie znamy dokładnie wymiarów zawartości znacznika, możemy mieć poważne kłopoty (rysunek 7.13). W przykładzie tym znajduje się wyróżniony cytat, mający za zadanie wyróżnić interesujący komentarz z artykułu. Szerokość i wysokość tego cytatu ustawiliśmy na 100 pikseli. Jeśli dodamy do niego zbyt dużo treści, to rozciągnie się ona na długość większą niż 100 pikseli i wyjdzie poza krawędź elementu. Nawet jeśli tekst mieści się w określonych ramach, to użytkownik może zwiększyć jego rozmiar w swojej przeglądarce.



Rysunek 7.13. *Gdy ustawisz wysokość elementu (jak w pasku bocznym na tym rysunku), ale jego treść będzie większa od tej wartości, to wyjdzie ona poza dolną krawędź swojego kontenera*

Innymi słowy, własność `height` jest przydatna do określania wysokości na przykład elementów `<div>` zawierających obrazy, ponieważ wysokość obrazu można łatwo sprawdzić. Jeśli jednak ustawisz wysokość elementowi zawierającemu tekst, sprawdź, jak to wygląda we wszystkich najważniejszych przeglądarkach i przy różnych ustawieniach rozmiaru pisma w tych programach.

Wskazówka: Kolejnym dobrym miejscem na wstawienie elementu o określonej wysokości jest obszar banera. Treść banera jest raczej przewidywalna: może zawierać na przykład logo, pole wyszukiwania albo kilka elementów nawigacyjnych. Często też banery zawierają dużo pustego miejsca (puste obszary przyciągają uwagę odwiedzających do najważniejszych elementów, np. paska nawigacyjnego), a więc określenie wysokości banera zazwyczaj nie powoduje problemów.

Przedefiniowywanie wymiarów pól

Napisałem wcześniej, że szerokość elementu jest sumą szerokości jego obramowania, dopełnienia i wartości `width`. Taki sposób obliczania szerokości elementów przez przeglądarki nie dość, że zmusza nas do wykonywania samodzielnych obliczeń, to jeszcze może powodować problemy. W szczególności dotyczy to układów tworzonych

w oparciu o elementy pływające i przy użyciu jednostek procentowych. Szczegółowo elementy pływające są opisane w dalszej części książki, ale krótko mówiąc, w CSS dostępna jest własność `float` umożliwiająca ustawianie elementów jeden obok drugiego, dzięki czemu można tworzyć kolumny.

Określając szerokość kolumn procentowo, można napotkać dziwne problemy. Powiedzmy, że mamy dwie kolumny (czyli np. dwa elementy `<div>`) i chcemy, aby każda z nich zajmowała 50 procent szerokości okna. Skoro tak, to przypisujemy im deklarację `width: 50%`. Jednak gdy takiej kolumnie zdefiniujemy dopełnienie lub obramowanie, to jej szerokość zwiększy się i będzie większa niż 50 procent (wyniesie 50% + suma szerokości lewego i prawego dopełnienia + suma szerokości lewej i prawej krawędzi obramowania). W większości przypadków efektem będzie brak wystarczającej ilości miejsca dla drugiej kolumny, która z tego powodu zostanie przeniesiona *pod* pierwszą kolumnę.

Na szczęście w CSS3 dodano nową własność pozwalającą określić, w jaki sposób przeglądarka ma obliczać szerokość i wysokość elementu na ekranie. Jest to własność `box-sizing`, która przyjmuje trzy wartości:

- Wartość `content-box` oznacza standardowy sposób obliczania wymiarów elementów, taki jak opisany wcześniej. Czyli szerokość i wysokość elementu są sumą wartości własności `width` i `height` oraz jego obramowania i dopełnienia. Jako że jest to domyślne ustawienie, nie trzeba go definiować, aby było stosowane.
- Wartość `padding-box` sprawia, że do szerokości i wysokości elementu ustawianych za pomocą własności `width` i `height` doliczane jest dopełnienie. Na przykład załóżmy, że lewej i prawe dopełnienie elementu ustawiliśmy na `20px`, a szerokość na `100px`. Dopełnienie zostanie potraktowane jako składnik szerokości i w rzeczywistości obszar treści w elemencie będzie miał szerokość 60 pikseli (100 i 20 [lewe dopełnienie] – 20 [prawe dopełnienie] = 60).
- Wartość `border-box` powoduje wliczenie w szerokość i wysokość dopełnienia i obramowania. To ustawienie rozwiązuje problem z opisywanymi wcześniej procentowymi wymiarami kolumn. Innymi słowy, gdy własność `box-sizing` zostanie ustawiona na `border-box`, to element o szerokości `50%` będzie zawsze zajmował połowę dostępnej przestrzeni, niezależnie od dopełnienia i obramowania.

Jeżeli nie podoba Ci się standardowy sposób obliczania szerokości i wysokości przez przeglądarki, korzystaj z własności `border-box` (chyba że masz jakiś niezwykły powód, aby do szerokości wliczyć tylko dopełnienie, bez obramowania). Sposób użycia własności `box-sizing` jest bardzo prosty. Wystarczy wpisać jej nazwę i po dwukropku podać jedną z trzech wartości:

```
box-sizing: border-box;
```

Pamiętaj jednak, że Firefox nie obsługuje jeszcze (w czasie pisania tej książki) standardowej nazwy tej własności, przez co trzeba dodać wersję z przedrostkiem (zobacz ramkę „Przedrostki producentów"). To samo dotyczy starszych wersji przeglądarki Safari dla systemu iOS i przeglądarki Android (do wersji 3 włącznie). Aby zapewnić działanie własności `box-sizing` we wszystkich tych przeglądarkach i innych, należy użyć trzech deklaracji:

```
-moz-box-sizing: border-box;
-webkit-box-sizing: border-box;
box-sizing: border-box;
```

Niektórym projektantom ustawienie `border-box` podoba się tak bardzo, że definiują je dla wszystkich elementów za pomocą selektora uniwersalnego:

```
* {
-moz-box-sizing: border-box;
-webkit-box-sizing: border-box;
box-sizing: border-box;
}
```

W rozdziale 14. opisuję też, jak bardzo ustawienie `border-box` jest przydatne przy tworzeniu responsywnych stron internetowych dopasowujących się rozmiarem do różnych urządzeń (tabletów, iPhone'a czy monitorów komputerów stacjonarnych).

Uwaga: Własność `box-sizing` działa nawet w przeglądarce Internet Explorer 8, a więc jest obsługiwana przez jakieś 95% wszystkich obecnie używanych przeglądarek. Niestety, nie rozpoznaje jej Internet Explorer 7, przez co w tej przeglądarce elementy z ustawieniem `border-box` mogą być znacznie szersze, niż planowałeś. Jeśli kierujesz swoje strony do odbiorców używających tej przeglądarki, niestety musisz zrezygnować z własności `box-sizing`.

Kontrolowanie wycieków za pomocą własności overflow

Jeśli treść elementu jest większa niż zdefiniowane w stylu wysokość i szerokość, to mogą mieć miejsce pewne nieprzewidywalne zdarzenia. Jak pokazano na rysunku 7.13, przeglądarki pozwalają treści wylać się na zewnątrz (za obramowanie i często na inną treść).

Na szczęście można sterować zachowaniem przeglądarki w takich sytuacjach za pomocą właściwości `overflow`. Przyjmuje ona jedną z czterech wartości w postaci słów kluczowych, które pozwalają zdefiniować sposób prezentacji treści wychodzącej poza granice pola:

- **`visible`** — określa normalne zachowanie przeglądarek. Jest równoznaczne z nieustawieniem właściwości `overflow` w ogóle (rysunek 7.14, na górze).
- **`scroll`** — dodaje paski przewijania (rysunek 7.14, na środku). Tworzy coś w rodzaju miniokna przeglądarki na stronie i wygląda podobnie do starych ramek HTML albo znacznika `<iframe>`. Tej opcji można użyć, aby zmieścić dużo treści na małej powierzchni. Niestety, paski przewijania pojawiają się zawsze, nawet wtedy, gdy treść nie przekracza rozmiaru bloku.
- **`auto`** — aby paski przewijania pojawiały się tylko opcjonalnie, należy użyć tej wartości. Ma taki sam efekt jak wartość `scroll`, ale dodaje paski przewijania tylko wtedy, gdy są potrzebne.
- **`hidden`** — ukrywa treść, która wychodzi poza blok (rysunek 7.14, na dole). Ta opcja jest nieco niebezpieczna, ponieważ może sprawić, że część zawartości strony zniknie. Jest jednak przydatna przy tworzeniu układów opartych na elementach pływających.

Rysunek 7.14. *Właściwość overflow pozwala kontrolować tekst przekraczający rozmiarem wymiary bloku na trzy sposoby. Wartość visible powoduje, że treść zawsze będzie widoczna (na górze), scroll i auto dodają paski przewijania (środek), a hidden ukrywa wszystko to, co wychodzi poza granice bloku (na dole)*

Określanie minimalnej szerokości i wysokości

Gdybyś jeszcze się nie zorientował, kaskadowe arkusze stylów to bardzo wszechstronne narzędzie. Dzięki temu oprócz standardowych własności `width` i `height` są dostępne jeszcze cztery inne, o podobnym przeznaczeniu:

- Własność **`max-width`**, jak można się spodziewać, służy do ustawiania maksymalnej szerokości elementów. Element taki może mieć mniejszą szerokość niż określona w tej własności, ale nie może jej przekroczyć. Przy użyciu tej własności można tworzyć elementy dostosowujące się do różnych szerokości ekranu, ale nie rozciągające się w nieskończoność na bardzo szerokich monitorach. Wyobraź sobie, że masz w arkuszu stylów poniższą regułę:

  ```
  body {
      max-width: 1200px;
  }
  ```

  Dzięki niej strona może dostosować się szerokością do mniejszych wyświetlaczy smartfonów czy tabletów, ale na szerokich monitorach jej szerokość nie przekroczy 1200 pikseli.

- Własność **`max-height`** działa podobnie do `max-width`, ale dotyczy wysokości. Przypomnę jednak to, co napisałem wcześniej: lepiej nie bawić się wysokością elementów.

- Własność **`min-width`** służy do określania minimalnej szerokości elementów. Element taki może mieć szerokość większą od ustawionej w tej własności, ale nie może być węższy. Jeśli przy zmniejszaniu szerokości okna przeglądarki zauważysz, że elementy robią się tak cienkie, że cały układ się rozpada, możesz im zdefiniować własność `min-width`:

  ```
  body {
      min-width: 760px;
  }
  ```

  Gdy użytkownik zmniejszy okno przeglądarki do szerokości mniejszej niż 760 pikseli, przeglądarka wyświetli poziomy pasek przewijania, zamiast dalej zmniejszać szerokość elementów.

- Własność **`min-height`** działa podobnie jak `min-width`, lecz dotyczy wysokości. Dzięki niej można rozwiązać problem pokazany na rysunku 7.13. Ustalając minimalną wysokość, informujemy przeglądarkę, że chcemy, aby element nie był niższy od pewnej wartości. Jeśli treść w tym elemencie jest wyższa od minimalnej wysokości, to cały element zostanie przez nią rozciągnięty.

Powyższych własności można używać razem. Przykładowo aby strona nigdy nie była węższa niż 760 pikseli i szersza niż 1200 pikseli, można zastosować poniższą regułę:

```
body {
    min-width: 760px;
    max-width: 1200px;
}
```

Elementy pływające

Normalnie elementy na stronie — nagłówki, akapity i inne elementy blokowe — układają się jeden nad drugim od góry do dołu. Ten znany z procesorów tekstu sposób prezentacji dokumentów jest niezbyt ekscytujący (rysunek 7.15, na górze). Dzięki CSS nie musimy jednak się tego trzymać. Mnóstwo metod układania elementów na stronach internetowych opisałem w trzeciej części tej książki. Jest jednak pewna właściwość, która w pojedynkę potrafi zdziałać naprawdę wiele — jest to właściwość `float`.

Właściwość `float` przenosi element albo w lewo, albo w prawo. W wyniku tego treść znajdująca się pod tym elementem przechodzi do góry i otacza go (rysunek 7.15, na dole). Elementy pływające są doskonałym sposobem na usunięcie dodatkowych informacji sprzed głównego tekstu na stronie. Obrazki mogą być przenoszone do każdej z krawędzi, co pozwala tekstowi elegancko je otaczać. Podobnie można też spowodować, aby pasek boczny zawierający powiązane ze sobą informacje i odnośniki znajdował się po prawej stronie.

Mimo że właściwość `float` jest wykorzystywana w wielu skomplikowanych (i trudnych do zrozumienia) zastosowaniach, o czym będzie mowa w rozdziale 13., jej podstawowa forma jest bardzo prosta. Jako wartość przyjmuje jedno z trzech słów kluczowych: `left`, `right` i `none`, na przykład:

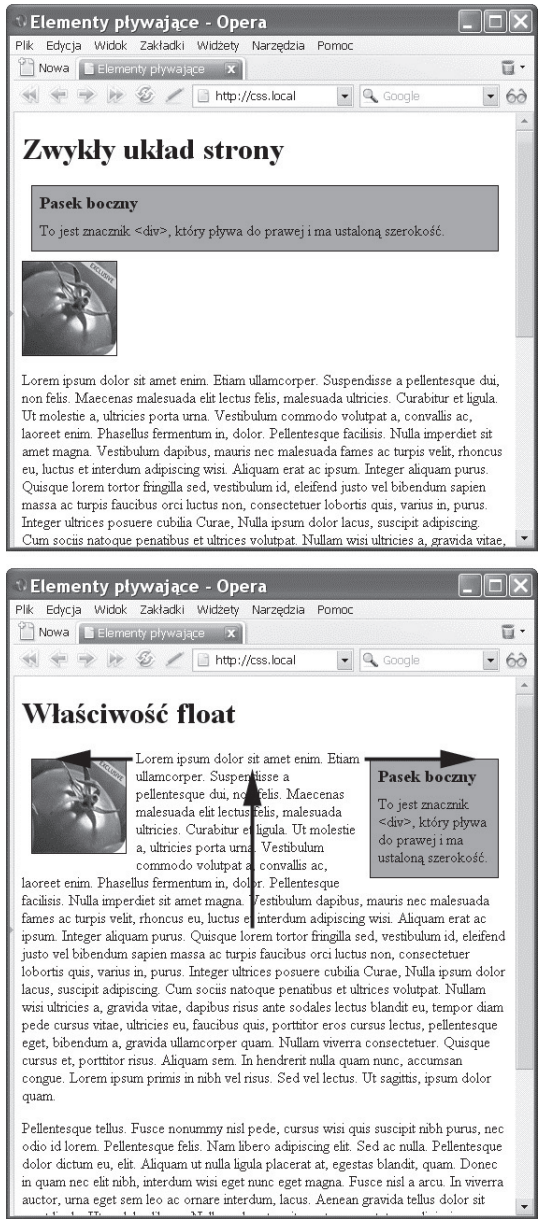
```
float: left;
```

- **`left`** — przenosi element na lewą stronę, a treść znajdująca się pod nim otacza go z prawej,
- **`right`** — przenosi element na prawą stronę,
- **`none`** — wyłącza pływanie i przywraca obiekt do jego normalnego położenia.

Elementy pływające dryfują do lewej lub prawej krawędzi okna przeglądarki albo *zawierającego je elementu*. Czasami tym zawierającym elementem jest okno przeglądarki, ale jeśli element pływający znajduje się w innym elemencie o ustalonej szerokości, to będzie przesunięty do lewej lub prawej krawędzie tego kontenera. Na przykład jeśli na stronie znajduje się blok o szerokości 300 pikseli, który dryfuje do prawej krawędzi okna przeglądarki, a w nim znajduje się obrazek dryfujący do lewej, to nie zostanie on przesunięty do lewej krawędzi okna, a do lewej krawędzi tego trzystupikselowego bloku.

Właściwość `float` można też stosować do elementów śródliniowych, takich jak znacznik `<img>`. Tak naprawdę spychanie obrazów do lewej lub prawej jest jednym z najczęstszych zastosowań właściwości `float`. Przeglądarki traktują elementy pływające jako blokowe, a więc nie ma problemów z dopełnieniem i marginesami, które normalnie występują w przypadku elementów śródliniowych.

Elementem pływającym może być każdy element blokowy, jak nagłówek czy akapit. Powszechnie spotykaną techniką jest stosowanie właściwości `float` do znacznika `<div>` (lub nowych elementów HTML, takich jak `<article>`, `<section>` czy `<aside>`) zawierającego inne znaczniki HTML i treść strony w celu utworzenia czegoś w rodzaju kontenera. W ten sposób można tworzyć paski boczne, cytaty



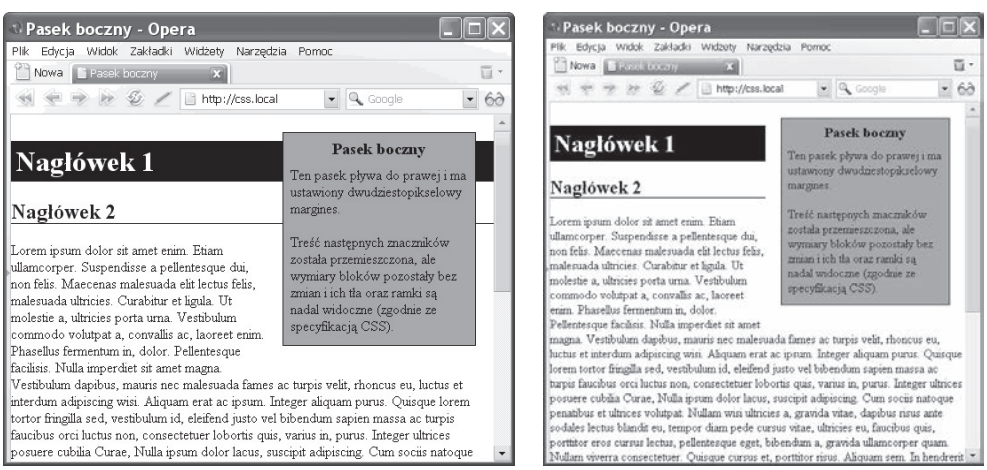
Rysunek 7.15. *Normalny przepływ HTML-a to od lewej do prawej i od góry do dołu i jeden element blokowy — nagłówek, akapit, znacznik itd. — znajduje się nad drugim. Dając możliwość wyrwania się z tej monotonii, właściwość float należy do najpotężniejszych i najbardziej przydatnych właściwości CSS. Znajduje ona zastosowanie od umieszczania obrazów po określonej stronie akapitu po kontrolę całych układów stron z banerami, paskami bocznymi i nawigacyjnymi oraz innymi elementami*

wyróżnione i inne elementy tworzące samodzielną całość (w kursie do tego rozdziału przedstawiłem tego przykład). Stosując właściwość `float` do elementów blokowych, dobrze jest też pamiętać o podaniu ich szerokości. W ten sposób można kontrolować ilość przestrzeni zajmowanej w poziomie przez blok oraz to, ile jej zostanie dla treści znajdującej się pod nim, która zostanie przeniesiona do góry i go otoczy.

Uwaga: *Kolejność znaczników* w pliku HTML ma bardzo duże znaczenie w przypadku stosowania elementów pływających. Znacznik elementu pływającego musi znajdować się przed znacznikami zawierającymi treść, która ma go otoczyć. Załóżmy, że mamy stronę, na której jest znacznik `<h1>`, a po nim `<p>`. Przy końcu tego drugiego znajduje się jeszcze obrazek. Jeśli obraz ten będzie pływał, powiedzmy, do prawej, to znacznik `<h1>` i większość zawartości znacznika `<p>` i tak będą znajdować się nad nim. Tylko to, co jest za znacznikiem pływającym, może go otaczać.

Tła i obramowanie a elementy pływające

Ku zdumieniu wielu webmasterów tła i obramowanie nie pływają w taki sam sposób jak inne elementy strony. Weźmy na przykład element stanowiący pasek boczny, który pływa do prawej. Treść znajdująca się pod nim przechodzi do góry i otacza go — zgodnie z oczekiwaniami. Jeśli jednak ta treść znajdowała się na jakimś tle lub miała obramowanie, to pojawią się one *pod* tym pływającym paskiem bocznym (rysunek 7.16, po lewej). W istocie przeglądarka oblewa pływający element tekstem, ale nie obramowaniem lub tłem. Możesz wierzyć lub nie, ale jest to absolutnie prawidłowe zachowanie (zgodne z zasadami). Możemy oczywiście nie chcieć stosować się do tych reguł, ponieważ wolelibyśmy, aby na przykład tło kończyło się przed elementem pływającym (rysunek 7.16, po prawej). Można to osiągnąć przy użyciu CSS.



Rysunek 7.16. *W tym przykładzie mamy znaczniki <h1> z określonym kolorem tła i <h2> z obramowaniem (po lewej). Dodanie reguły overflow: hidden; do stylu znacznika <h1> (po prawej) zapobiega ukazywaniu się tła nagłówka pod elementem pływającym (paskiem bocznym)*

Najpierw trzeba dodać do stylu tworzącego obramowanie lub tło jedną regułę. Gdy znajdziesz już odpowiedni styl, dodaj do niego ten wiersz: `overflow: hidden;`. Dzięki właściwości `overflow` tło lub obramowanie wychodzące poza element nie będą pokazywane.

Innym sposobem jest ustawienie obramowania wokół elementu pływającego — jeśli utworzymy wystarczająco grube obramowanie o takim samym kolorze jak tło strony, to będzie ono wyglądało jak pusta przestrzeń, mimo że w rzeczywistości będzie tylko zasłaniać i ukrywać kolor tła oraz obramowania wychodzące poza ten element pływający.

Pływaków nie wpuszczamy

Czasami trzeba sprawić, aby jakiś znacznik ignorował to, że niektóre elementy są pływające. Możemy na przykład chcieć, aby informacja o prawach autorskich znajdowała się zawsze na dole okna przeglądarki. Gdybyśmy mieli na stronie bardzo wysoki pasek boczny pływający do lewej, to notka o prawach autorskich mogłaby zostać przeniesiona do góry i owinięta wokół tego elementu pływającego. My natomiast chcemy, aby informacje o prawach autorskich na stronie nie otaczały tego elementu pływającego, tylko znajdowały się pod nim.

Inny problem powstaje w sytuacji, gdy obok siebie znajduje się kilka elementów pływających. Jeśli nie są one zbyt szerokie, to wszystkie zostaną umieszczone obok siebie w jednej linii, i jeśli mają różne wysokości, to mogą utworzyć niezbyt atrakcyjny wizualnie obraz (rysunek 7.17, na górze). W takim przypadku elementy pływające *nie powinny* pływać obok siebie. Do rozwiązywania tego typu problemów w CSS służy właściwość `clear`.

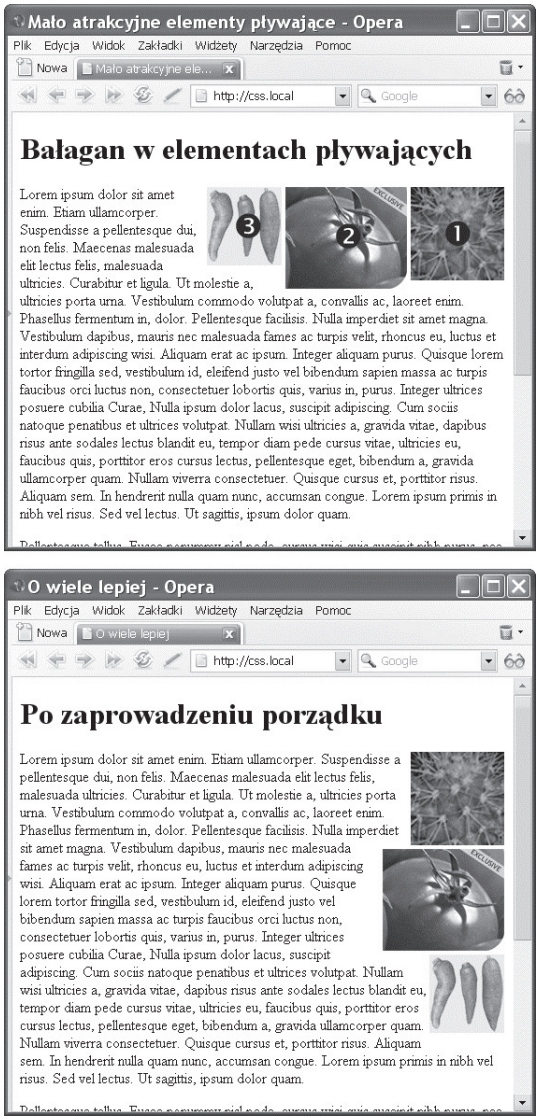
Właściwość `clear` *zapobiega owijaniu się* elementu wokół elementu pływającego. Stosując ją do elementu, w istocie zmusza się go do przeniesienia pod element pływający. Można określić, którego rodzaju elementy pływające (do prawej czy do lewej) mają być ignorowane lub nakazać ignorowanie każdego ich typu.

Właściwość `clear` może przyjmować następujące wartości:

- **`left`** — element będzie przenoszony pod elementy pływające do lewej, ale będzie się zachowywał normalnie wobec tych, które pływają do prawej.
- **`rigth`** — element będzie przechodził pod elementy pływające do prawej, ale będzie się owijał wokół elementów pływających do lewej.
- **`both`** — element będzie przechodził pod wszystkie elementy pływające.
- **`none`** — całkiem wyłącza właściwość `clear`. Innymi słowy, sprawia, że element będzie się owijał wokół elementów pływających do prawej, jak i do lewej, co jest normalnym zachowaniem zdefiniowanym w przeglądarkach.

Notka o prawach autorskich powinna zawsze znajdować się na samym dole strony, a więc powinniśmy do jej znacznika zastosować właściwość `clear` z wartością, która wyczyści elementy pływające po obu stronach, czyli `both`. Dzięki temu notka będzie zawsze znajdowała się pod wszystkimi innymi obiektami i nie będzie się owijał wokół żadnego z nich. Poniższa klasa wykonuje to zadanie:

```
.copyright {
    clear: both;
}
```



Rysunek 7.17. *Na górze: czasami nie chcemy, aby dany element znajdował się obok elementu pływającego. Na dole: zastosowanie właściwości clear (w tym przypadku z wartością right) do każdego obrazu zapobiega ich układaniu się w jednej linii. Zdjęcie 2 nie pojawia się już obok 1, a zdjęcie 3 zostało przeniesione pod 2*

Na rysunku 7.17 widać, w jaki sposób właściwość `clear` zapobiega tłoczeniu się różnej wysokości elementów pływających. Wszystkie obrazki na tym rysunku pływają do prawej. Na górze zdjęcie pomidorów (1) pojawia się jako pierwsze w pliku HTML i jest najdalej z wszystkich wysunięte w prawo. Drugie zdjęcie (2), biorąc pod uwagę to, że poprzednie pływa do prawej, ułożyło się po jego lewej stronie. To samo stało się ze zdjęciem trzecim (3).

Zastosowanie właściwości `clear: right;` spowoduje, że zdjęcia nie będą układały się w jednej linii jedno obok drugiego (rysunek 7.17, na dole). Właściwość `clear` zastosowana do drugiego zdjęcia zapobiega jego przeniesieniu w miejsce po lewej stronie zdjęcia pierwszego, a trzecie przenosi pod drugie.

Wskazówka: Reguły dotyczące elementów pływających i stosowania właściwości `clear` wydają się skomplikowane. Ten rozdział jest tylko wprowadzeniem do ich stosowania. Do tego tematu wrócimy jeszcze w rozdziale 13., gdzie nauczymy się stosować je w bardziej wyszukany sposób.

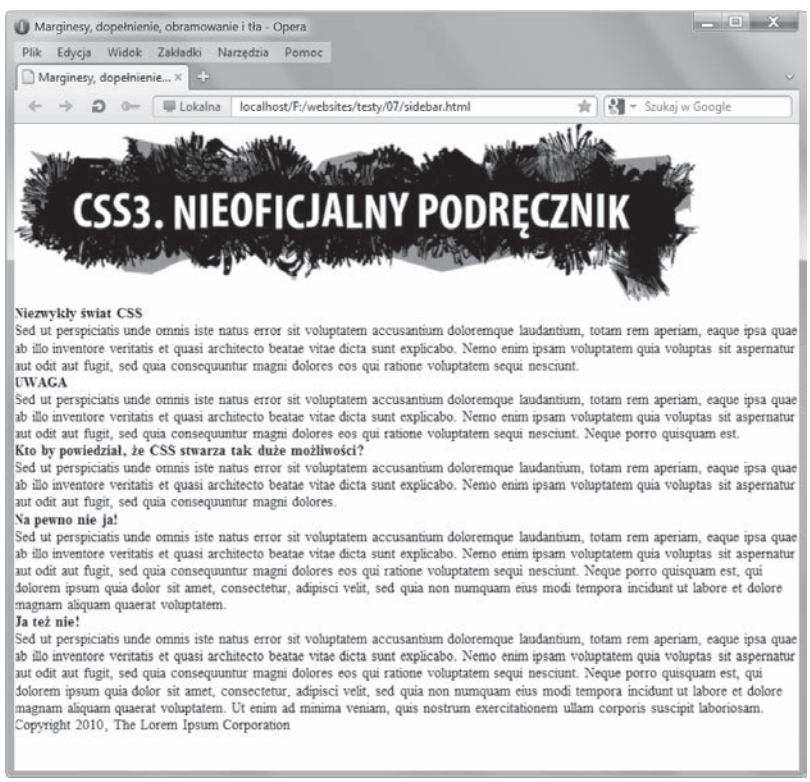
Kurs: marginesy, tła i obramowanie

W tym kursie zajmiemy się elementami modelu polowego CSS, ustawianiem odstępów wokół obiektów na stronie, dodawaniem kolorowych ramek oraz kontrolowaniem rozmiaru i ułożenia elementów strony.

1. **Do rozpoczęcia pracy potrzebne będą pliki dostępne pod adresem** *www.helion.pl/ksiazki/css3n3.htm***.** Pobierz te pliki i, jako że są one spakowane w archiwum ZIP, rozpakuj je. Pliki dotyczące tego kursu znajdują się w folderze o nazwie *r07*.

Ustawianie tła i marginesów

Pracę rozpoczniemy od prostego pliku HTML zawierającego wewnętrzny arkusz stylów z prostym resetem CSS. Na razie nie ma się czym chwalić (rysunek 7.18).



Rysunek 7.18. *Ta strona to prosty dokument HTML zawierający tylko jedną regułę CSS usuwającą większość domyślnych ustawień formatowania elementów. Gdy skończymy pracę nad nią, będzie wyglądała o wiele lepiej*

Wskazówka: Rezultat, jaki powinniśmy osiągnąć, można podejrzeć na rysunku 7.21.

1. **Otwórz w edytorze plik *r07/sidebar.html*.**

   Strona ta zawiera już arkusz stylów z tymi samymi regułami co opisane w rozdziale 5. Mówiąc krótko, style te usuwają wszystkie marginesy, dopełnienie i ujednolicają rozmiar pisma w najczęściej używanych elementach blokowych oraz eliminują wiele problemów związanych z niespójną prezentacją elementów w różnych przeglądarkach.

   Style te powinno się umieszczać w każdym arkuszu stylów. Najważniejsze ustawienia dotyczą marginesów i dopełnienia — zostały one zdefiniowane w pierwszej regule. Własności te są tak pogmatwane w różnych przeglądarkach, że najlepiej jest je wyzerować i zdefiniować od nowa. Na początek zdefiniujemy coś łatwego — kolor tła.

2. W arkuszu stylów kliknij za komentarzem resetu CSS (`/* */`) i wpisz poniższą regułę:

   ```
   html {
       background-color: rgb(253,248,171);
   }
   ```

   Reguła ta ustawia żółty kolor tła całej strony. Aby ustawić kolor tła całej strony, można zastosować własność `background-color` do elementu `<html>` lub `<body>`. Teraz zdefiniujemy marginesy, obramowanie i kilka innych ustawień dla elementu `<body>`.

Uwaga: Możliwe, że jesteś przyzwyczajony do szesnastkowego formatu zapisu wartości kolorów (typu `#FDF8AB`), a nie do RGB. Jeśli tak, to pod adresem *www.colorhexa.com/* znajduje się narzędzie służące do konwersji między tymi dwoma formatami. Format RGB ma tę przewagę nad szesnastkowym, że występuje też w wersji RGBA z obsługą przezroczystości (rozdział 6.), a jeśli już używa się jednego formatu w jakimś miejscu, to najlepiej stosować go wszędzie.

3. **Dodaj poniższą regułę do arkusza stylów:**

   ```
   body {
       background-color: rgb(255,255,255);
       border: 3px solid rgb(75,75,75);
   }
   ```

   Ta reguła ustawia biały kolor tła i 3-pikselowe ciemnoszare obramowanie dla elementu `<body>`. Jako że element `<body>` znajduje się w elemencie `<html>`, przeglądarka traktuje go tak, jakby znajdował się na nim. W związku z tym biały kolor przykryje wcześniej zdefiniowany żółty. W następnej kolejności określimy szerokość elementu `<body>` oraz ustawimy mu marginesy i dopełnienie.

Wskazówka: Normalnie tło elementu `<body>` pokrywa całą powierzchnię okna przeglądarki. Jeśli jednak zdefiniuje się też kolor dla elementu `<html>`, to tło elementu `<body>` będzie pokrywać tylko obszar treści. Aby zobaczyć, jak to wygląda, obejrzyj stronę po wykonaniu czynności opisanych w punkcie 3. Następnie usuń regułę dotyczącą elementu `<html>` i odśwież stronę. To niezwykła, ale przydatna sztuczka.

4. **Dodaj do reguły elementu <body> pięć kolejnych deklaracji (zmiany zostały wyróżnione):**

```
body {
    background-color: rgb(255,255,255);
    border: 3px solid rgb(75,75,75);
    width: 760px;
    margin-top: 20px;
    margin-left: auto;
    margin-right: auto;
    padding: 15px;
}
```

Własność `width` ustawia szerokość elementu na 760 pikseli. Jeśli okno przeglądarki odwiedzającego będzie szersze, to po bokach ukaże się kolorowe tło elementu `<html>`, na którym będzie się znajdowało 760-pikselowe białe pole, będące elementem `<body>`.

Własność `margin-top` odsuwa element `<body>` na odległość 20 pikseli od górnej krawędzi okna przeglądarki, natomiast wartości lewego i prawego marginesu powodują ustawienie tego elementu na środku okna przeglądarki w poziomie. Wartość `auto` oznacza, że przeglądarka ma sama określić rozmiar marginesu, a ponieważ została ona zastosowana z dwóch stron, przeglądarka ustawia oba marginesy na jednakową wartość.

Uwaga: Powyższe trzy wiersze kodu dotyczące marginesów można skompresować w jednej własności zbiorczej `margin` (rozdział 6.):

```
margin: 20px auto 0 auto;
```

Ostatnia własność odsuwa zawartość elementu `<body>` od jego krawędzi na odległość 15 pikseli. Inaczej mówiąc, dzięki własności `padding: 15px` obraz i tekst są oddalone od wszystkich czterech krawędzi swojego kontenera o 15 pikseli. Teraz utworzymy poświatę wokół elementu `<body>` za pomocą własności `box-shadow`.

5. **Do reguły elementu <body> dodaj jeszcze jedną deklarację. Wstaw ją za deklaracją obramowania, a przed deklaracją szerokości (zmiany są wyróżnione):**

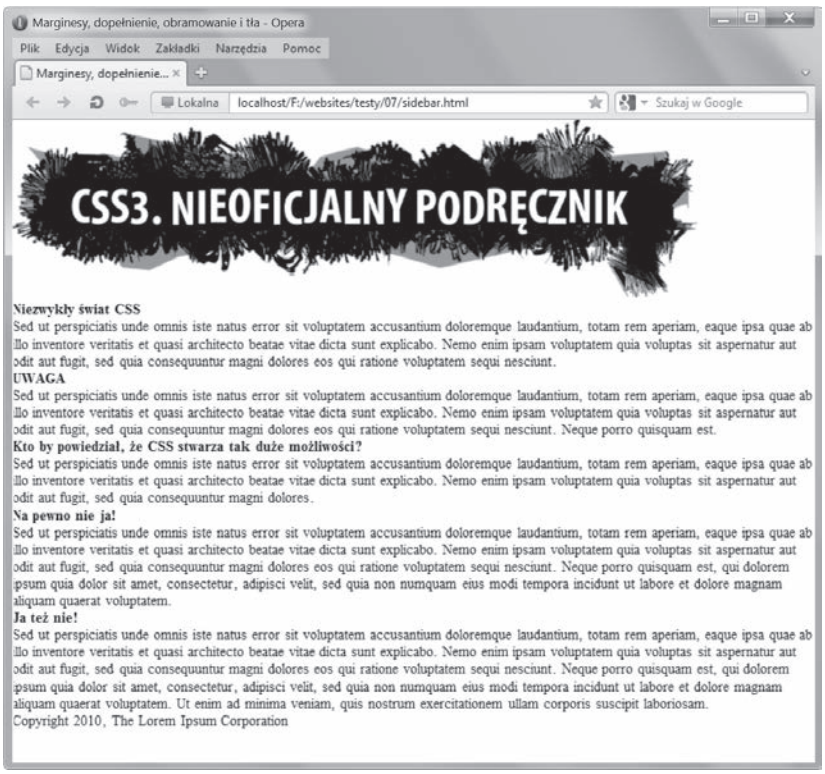
```
body {
    background-color: rgb(255,255,255);
    border: 3px solid rgb(75,75,75);
    box-shadow: 0 0 15px 5px rgba(44,82,100,.75);
    width: 760px;
    margin-top: 20px;
    margin-left: auto;
    margin-right: auto;
    padding: 15px;
}
```

Ta dodatkowa deklaracja tworzy wokół elementu 15-pikselowy cień tworzący poświatę (znajdujące się na początku wartości `0 0` oznaczają, że cień ma nie być przesunięty w żadnym kierunku). Wartość `5px` określa rozmiar cienia i wypycha go na odległość pięciu pikseli wokół wszystkich czterech krawędzi. Na końcu znajduje się definicja RGBA częściowo przezroczystego ciemnoniebieskiego koloru.

Dodane deklaracje dają już jakiś efekt, który można zaobserwować, odświeżając stronę.

6. **Zapisz stronę i obejrzyj efekty swojej pracy w przeglądarce.**

   Na stronie powinno znajdować się białe pole z obrazem, kilkoma akapitami tekstu i szarym obramowaniem z niebieskawą poświatą, a całość powinna znajdować się na żółtym tle (rysunek 7.19). Tekst wymaga naszej troski. Zajmiemy się nim teraz.



Rysunek 7.19. *Ustawienie lewego i prawego marginesu elementu o określonej szerokości na auto powoduje jego wyśrodkowanie. W tym przypadku ustawienia margin-left: auto i margin-right: auto zastosowane zostały do elementu <body>, który dzięki temu został ustawiony na środku okna przeglądarki. Niestety, nie ma łatwego sposobu na wyśrodkowanie elementu w pionie (tzn. sprawienie, aby nad i pod nim było tyle samo wolnej przestrzeni), chociaż bystrzy projektanci radzą sobie z tym problemem, stosując różne sprytne sztuczki. Jeśli chcesz się dowiedzieć, jak wyśrodkować element w pionie, zajrzyj na stronę* www.vanseodesign.com/css/vertical-centering/

Ustawianie odstępów wokół znaczników

Jako że reset CSS, który zastosowaliśmy na tej stronie, pozbawił tekst elementów praktycznie wszelkiego formatowania, aby nagłówki i akapity dobrze wyglądały, musimy je od nowa zdefiniować. Zaczniemy od upiększenia znajdującego się na górze elementu `<h1>`.

1. **Wróć do pliku *sidebar.html* w edytorze**. Kliknij za zamykającą klamrą reguły elementu `<body>`, naciśnij klawisz *Enter*, aby przejść do nowego wiersza, i wpisz poniższą regułę:

```
h1 {
    font-size: 2.75em;
    font-family: Georgia, "Times New Roman", Times, serif;
    font-weight: normal;
    text-align: center;
    letter-spacing: 1px;
    color: rgb(133,161,16);
    text-transform: uppercase;
}
```

W regule tej zostało użytych wiele własności formatowania tekstu opisanych w poprzednim rozdziale — teraz nagłówek ma rozmiar `2.75em` (co w większości przeglądarek będzie równoznaczne z 44 pikselami), jest drukowany w całości wielkimi literami, ma krój czcionki Georgia, jest zielony, a litery są delikatnie rozstrzelone. Własność `text-align` wyśrodkowuje tekst w zawierającym go polu. Jednak najciekawszym efektem będzie dodanie tła, które mocno wyróżni nagłówek na tle reszty strony.

Wskazówka: Po wykonaniu czynności opisanych w każdym punkcie zapisz stronę i podejrzyj ją w przeglądarce. Dzięki temu lepiej zrozumiesz, w jaki sposób działają poszczególne własności CSS.

2. **Dodaj kolejną (wyróżniona) własność do poprzedniej reguły**:

```
h1 {
    font-size: 2.75em;
    font-family: Georgia, "Times New Roman", Times, serif;
    font-weight: normal;
    text-align: center;
    letter-spacing: 1px;
    color: #85A110;
    text-transform: uppercase;
    background-color: rgb(226,235,180);
}
```

Teraz nagłówek ma jasnozielone tło. W elementach blokowych tło pokrywa całą powierzchnię i dlatego widać je od lewej do prawej krawędzi. Inaczej mówiąc, kolorowe tło nie jest wyświetlone tylko pod tekstem (tu: **Niezwykły świat CSS**), ale i na powierzchni całego elementu.

Jednak tekst w tym nagłówku ma trochę zbyt mało przestrzeni. Aby to poprawić, użyjemy dopełnienia.

3. **Dodaj kolejną deklarację do reguły elementu `<h1>`. Reguła ta powinna teraz wyglądać tak, jak poniżej (zmiany zostały wyróżnione)**:

```
h1 {
    font-size: 2.75em;
    font-family: Georgia, "Times New Roman", Times, serif;
    font-weight: normal;
    text-align: center;
    letter-spacing: 1px;
    color: #85A110;
    text-transform: uppercase;
    background-color: rgb(226,235,180);
    padding: 5px 15px 2px 15px;
}
```

Własność zbiorcza `padding` pozwala w zwięzły sposób zdefiniować dopełnienie z wszystkich czterech stron elementu. W tym przypadku zostały zdefiniowane następujące ustawienia dopełnienia: `5px` u góry, `15px` z prawej, `2px` u dołu i `15px` z lewej.

Jest jeszcze jeden problem, który trzeba rozwiązać. W sekcji „Ustawianie tła i marginesów" w punkcie 5., zostało zdefiniowane dopełnienie dla elementu `<body>`, przez co nagłówek jest odsunięty na odległość 15 pikseli od lewej i prawej krawędzi tego elementu. Lepiej by jednak wyglądał, gdyby stykał się z tymi zielonymi krawędziami. Problem ten można łatwo rozwiązać przy użyciu ujemnych marginesów.

4. **Dodaj jeszcze jedną deklarację do reguły `h1`, aby wyglądała tak, jak poniżej (zmiany są wyróżnione):**

```
h1 {
    font-size: 2.75em;
    font-family: Georgia, "Times New Roman", Times, serif;
    font-weight: normal;
    text-align: center;
    letter-spacing: 1px;
    color: #85A110;
    text-transform: uppercase;
    background-color: rgb(226,235,180);
    padding: 5px 15px 2px 15px;
    margin: 0 -15px 20px -15px;
}
```

W tej regule została zastosowana zbiorcza własność `margin`, za pomocą której ustawiliśmy górny margines na `0`, prawy i lewy na `-15px`, a dolny na `20px`. Dolny margines tworzy wolną przestrzeń między nagłówkiem i znajdującym się pod nim akapitem. Najciekawsze w tej regule są ujemne wartości dla prawego i lewego marginesu. Jak napisałem w rozdziale 6., ujemny margines można zdefiniować każdemu elementowi. Margines taki rozciąga element w określonym kierunku — w tym przypadku nagłówek jest rozciągany w lewo i prawo o 15 pikseli, dzięki czemu wchodzi na dopełnienie zawierającego go kontenera.

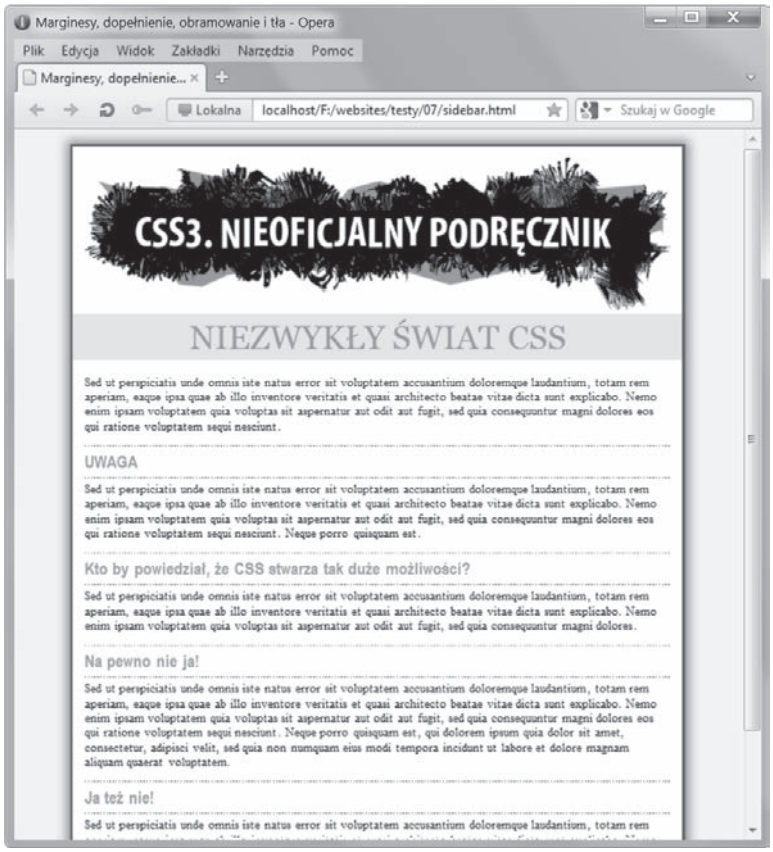
5. **Teraz zdefiniujemy formatowanie elementów `h2`**. Wpisz poniższą regułę pod regułą `h1`:

```
h2 {
    font-size: 1.5em;
    font-family: "Arial Narrow", Arial, Helvetica, sans-serif;
    color: rgb(249,107,24);
    border-top: 2px dotted rgb(141,165,22);
    border-bottom: 2px dotted rgb(141,165,22);
    padding-top: 5px;
    padding-bottom: 5px;
    margin: 15px 0 5px 0;
}
```

Deklaracje znajdujące się w tej regule określają podstawowe właściwości formatowania tekstu oraz tworzą przerywaną linię pod elementami `<h2>`. Dodatkowo zdefiniowane zostało niewielkie górne i dolne dopełnienie, mające za zadanie odsunąć tekst nagłówka od jego krawędzi. Znajdująca się na końcu deklaracja marginesów odsuwa nagłówek o 15 pikseli od elementu znajdującego się nad nim i o 5 pikseli od elementu znajdującego się pod spodem.

6. **Zapisz plik i obejrzyj efekt swoich działań w przeglądarce**.

Teraz nagłówki wyglądają całkiem dobrze (rysunek 7.20). W następnej kolejności utworzymy pasek boczny po prawej stronie dokumentu.



Rysunek 7.20. *Wystarczyło kilka deklaracji CSS, aby zdefiniować kolory tła, marginesy oraz odstępy między akapitami i nagłówkami*

Tworzenie paska bocznego

Paski boczne są powszechnie spotykane w publikacjach drukowanych, takich jak czasopisma, książki czy gazety. Zawierają one i wyróżniają małe porcje informacji, takie jak listy źródeł, informacje kontaktowe czy związane z opisywanym tematem dygresje. Jednak aby były efektywne, paski boczne nie mogą zaburzać naturalnego układu strony. Powinny, jak wskazuje ich nazwa, pływać sobie przy którejś z krawędzi, nie wchodząc „w drogę" reszcie strony. Efekt ten można z łatwością osiągnąć przy użyciu CSS.

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

   Najpierw trzeba wyodrębnić obszar, który będzie zajmowany przez pasek boczny. Do tego zadania idealnie nadaje się znacznik `<div>`. Można w nim umieścić dowolną ilość kodu HTML stanowiącego odrębną całość na stronie.

2. **Przewiń w dół do kodu HTML i kliknij przed *pierwszym* znacznikiem `<h2>` (zawierającym tekst *UWAGA*).** Następnie wpisz `<div class="sidebar">` i naciśnij klawisz *Enter*.

Ten kod oznacza początek paska bocznego i stosuje do niego klasę `sidebar`. Styl tej klasy utworzymy w następnym punkcie, ale przedtem musimy jeszcze zaznaczyć koniec paska, zamykając znacznik `<div>`.

3. **Kliknij za zamykającym znacznikiem `</p>` znajdującym się za elementem `<h2>` (jest to znacznik `<p>` znajdujący się bezpośrednio przed elementem `<h2>Kto by powiedział, że CSS stwarza tak duże możliwości?</h2>`).** Naciśnij klawisz *Enter* i wpisz `</div>`.

   Właśnie umieściliśmy nagłówek i akapit wewnątrz znacznika `<div>`. Teraz utworzymy dla niego styl.

4. **Przewiń z powrotem do góry do arkusza stylów i dodaj poniższy styl pod wcześniej utworzonym stylem znacznika `<h2>`:**

```
.sidebar {
    width: 30%;
    float: right;
    margin: 10px;
}
```

   Ten styl ustawia szerokość obszaru treści (tego, w którym znajduje się tekst) na 30 procent. Nie trzeba było używać bezwzględnej jednostki, takiej jak piksele — szerokość paska bocznego będzie wynosić 30 procent szerokości jego kontenera. Własność `float` przenosi pasek na prawą stronę, a własność `margin` ustawia wokół niego 10-pikselową wolną przestrzeń.

   Jeśli obejrzymy tę stronę w przeglądarce teraz, to zobaczymy, że podstawowy kształt i położenie paska bocznego zostały już ustawione, ale jest jeden problem: obramowanie znacznika `<h2>` jest widoczne *pod* paskiem. Podczas gdy tekst nagłówków został przesunięty, tła i obramowanie zostały tam, gdzie były. Te elementy pozostały pod paskiem. Problem ten można rozwiązać, definiując paskowi bocznemu tło, dzięki któremu nie będzie widać obramowania elementów `<h2>` (jest jeszcze inna możliwość, opisana w punkcie 8.).

5. **Dodaj do klasy `style` deklaracje wyróżnione poniżej:**

```
.sidebar {
    width: 30%;
    float: right;
    margin: 10px;
    background-color: rgb(250,235,199);
    padding: 10px 20px;
}
```

   Deklaracje te definiują jasnopomarańczowy kolor tła oraz odsuwają tekst od krawędzi elementu, które za chwilę również zostaną zdefiniowane.

6. **Dodaj dwie kolejne deklaracje do stylu `.sidebar` (zmiany zostały wyróżnione):**

```
.sidebar {
    width: 30%;
    float: right;
    margin: 10px;
    background-color: rgb(250,235,199);
    padding: 10px 20px;
    border: 1px dotted rgb(252,101,18);
    border-top: 20px solid rgb(252,101,18);
}
```

Jest to przykład wykorzystania techniki opisanej we wcześniejszej części rozdziału. Jeśli chcesz, aby trzy krawędzie obramowania elementu były takie same, a jedna się od nich różniła, możesz najpierw zdefiniować styl dla wszystkich czterech krawędzi — tu: przerywana pomarańczowa linia o grubości jednego piksela — a następnie zmienić jedną z nich — tu: górna krawędź została zmieniona na 20 pikseli grubości i linię ciągłą. Dzięki tej metodzie wystarczy napisać tylko dwa wiersze kodu CSS zamiast czterech (`border-top`, `border-right`, `border-bottom` i `border-left`).

Teraz upiększymy nasz boczny pasek, zaokrąglając jego rogi i dodając mu cień.

7. **Dodaj jeszcze dwie własności do reguły `.sidebar`, aby wyglądała jak poniżej (zmiany zostały wyróżnione):**

```
.sidebar {
    width: 30%;
    float: right;
    margin: 10px;
    background-color: rgb(250,235,199);
    padding: 10px 20px;
    border: 1px dotted rgb(252,101,18);
    border-top: 20px solid rgb(252,101,18);
    border-radius: 10px;
    box-shadow: 5px 5px 10px rgba(0,0,0,.5);
}
```

Własność `border-radius` służy do zaokrąglania rogów. Zastosowane w tym przypadku ustawienie `10px` tworzy wyróżniające się wydatne zakrzywienie. Własność `box-shadow` definiuje cień rzucany po prawej stronie i u dołu elementu, dzięki czemu wygląda on tak, jakby unosił się nieco nad stroną. Prawie skończyliśmy.

Nagłówek w pasku bocznym nie wygląda jeszcze dokładnie tak, jak powinien — ma te same właściwości co pozostałe elementy `<h2>` na stronie (odnosi się do niego selektor `h2` utworzony w punkcie 4.). Obramowanie przeszkadza, a górny margines spycha nagłówek za bardzo w dół. Można to poprawić przy użyciu selektora potomka.

8. **Dodaj za regułą `.sidebar` poniższą regułę z selektorem potomka:**

```
.sidebar h2 {
    border: none;
    margin-top: 0;
    padding: 0;
}
```

Dzięki klasie `.sidebar` ten styl jest silniejszy — tzn. bardziej *precyzyjny* — od prostego selektora `h2`. Reguła ta usuwa obramowanie, górny margines oraz dopełnienie ze wszystkich stron. Natomiast ustawienia rozmiaru, koloru i kroju pisma pozostają niezmienione, ponieważ nie zdefiniowaliśmy ich tutaj — tak działa kaskada!

Strona wygląda już całkiem dobrze, ale obramowanie elementów `h2` wciąż wchodzi pod pasek boczny. Nie wygląda to dobrze, jednak można temu łatwo zaradzić.

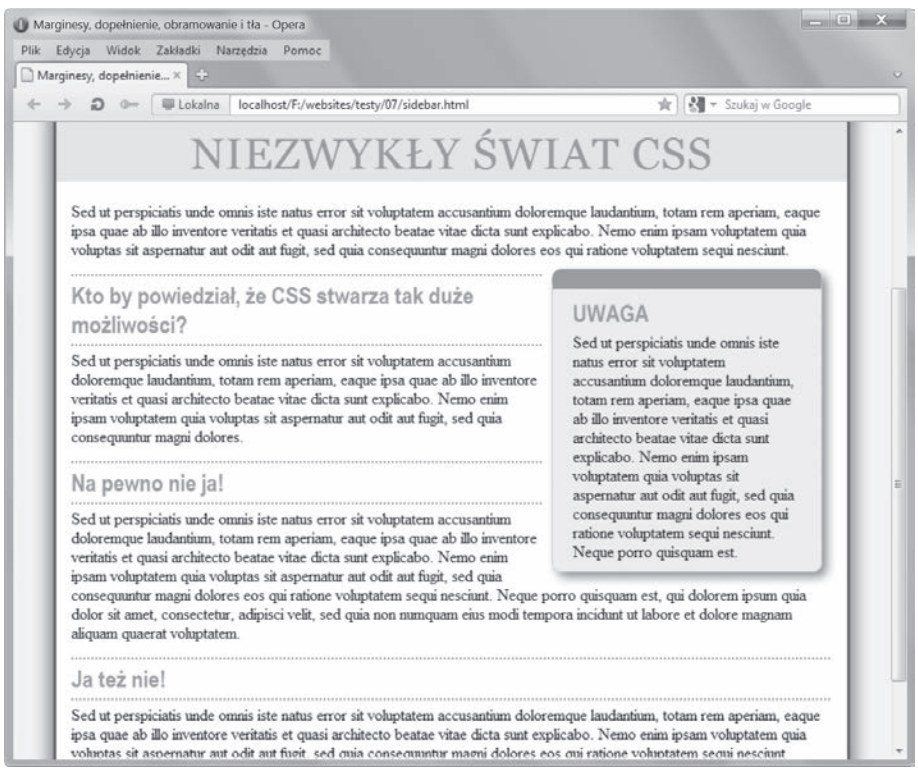
9. **Odszukaj regułę z selektorem `h2` i dodaj do niej deklarację własności `overflow`:**

```
h2 {
    font-size: 1.5em;
    font-family: "Arial Narrow", Arial, Helvetica, sans-serif;
    color: rgb(249,107,24);
    border-top: 2px dotted rgb(141,165,22);
    border-bottom: 2px dotted rgb(141,165,22);
    padding-top: 5px;
    padding-bottom: 5px;
    margin: 15px 0 5px 0;
    overflow: hidden;
}
```

Ustawienie własności `overflow` na `hidden` sprawia, że obramowanie wykraczające poza tekst i wchodzące pod pasek boczny zostanie ukryte.

10. **Zapisz plik i otwórz stronę w przeglądarce.**

    Strona powinna wyglądać tak, jak na rysunku 7.21.



Rysunek 7.21. *Kilka stylów CSS przeistacza nudny szary HTML w elegancką stronę. Zwróć uwagę na to, jak pasek boczny przyciąga uwagę, a zarazem odciąga ją od głównego tekstu strony*

O krok dalej

Aby wypróbować nowo nabyte umiejętności, wykonaj samodzielnie następujące ćwiczenie: utwórz regułę z selektorem `p` i zdefiniuj w niej trochę bajerów dla akapitów, na przykład pokombinuj z ustawieniami marginesów, kolorem pisma itd.

Następnie utwórz styl klasy formatujący informację o prawach autorskich, która znajduje się na samym dole strony *sidebar.html* (niech się nazywa, powiedzmy, `.copyright`). W tym stylu utwórz górną krawędź nad notką, zmień kolor jej tekstu, zmniejsz rozmiar pisma i spraw, aby tekst był pisany wielkimi literami (podpowiedź: użyj właściwości `text-transform` opisanej w rozdziale 6.). Po utworzeniu stylu zastosuj go do akapitu z notką o prawach autorskich.

Skorowidz

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

Ś

T

U

W

X

Z

NOTATKI

CSS3

nieoficjalny podręcznik

Kaskadowe arkusze stylów (ang. Cascading Style Sheets, CSS) odmieniły świat stron internetowych. Pozwoliły oddzielić treść strony od jej wyglądu. Od tej chwili życie projektantów stało się zdecydowanie prostsze. Praca nad zmianą wyglądu strony przestała przyprawiać o zawrót głowy, a szybkie przygotowanie kilku wersji na konkretne urządzenia stało się realne. Ostatnia wersja CSS3 wprowadza kolejne nowości, które zapewnią optymalne wykorzystanie potencjału stron WWW.

Ta książka należy do cenionej serii ***Nieoficjalny podręcznik***. W trakcie lektury dowiesz się, jak przygotować poprawny arkusz stylów i dołączyć go do strony. Poznasz najlepsze techniki formatowania konkretnych elementów kodu HTML. Ponadto przekonasz się, że można szybko i wygodnie zaprojektować profesjonalną stronę WWW. Znajdziesz tu również opis nowości HTML-a w wersji 5 oraz nauczysz się korzystać z innych czcionek niż Arial czy Verdana. Dzięki CSS3 wiele operacji, które do tej pory sprawiały kłopoty lub wymagały korzystania z elementów graficznych, uda Ci się wykonać za pomocą zaledwie paru linijek kodu. Zaokrąglone rogi, obracanie elementów, gradienty to tylko niektóre z nich. Książka ta jest obowiązkową lekturą dla każdego projektanta i webmastera chcącego być na bieżąco z nowinkami ze świata stron WWW.

Poznaj sekrety CSS!

- Formatowanie stron
- Wykorzystanie niestandardowych czcionek
- Obracanie elementów
- Animacje

Najlepszy przewodnik po CSS3 dla webmasterów!

helion.pl księgarnia internetowa

Nr katalogowy: 14663

Księgarnia internetowa
http://helion.pl

Zamówienia telefoniczne:
0 801 339900
0 601 339900

O'REILLY

Helion

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

sięgnij po WIĘCEJ
KOD KORZYŚCI

ISBN 978-83-246-7317-9

9 788324 673179

Cena: 79,00 zł