

Biblia webmastera!

CSS3

nieoficjalny podręcznik

Wydanie III



David Sawyer McFarland

O'REILLY



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

Tłumaczenie: Łukasz Piwko

ISBN: 978-83-246-7317-9

© 2013 Helion S.A.

Authorized Polish translation of the English edition of CSS3: The Missing Manual, 3rd Edition, ISBN 9781449325947 © 2013 Sawyer McFarland Media, Inc.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

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.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Nieoficjalna czołówka	15
Wstęp	19
Cześć I. Podstawy CSS	31
Rozdział 1. Przystosowanie kodu HTML do pracy z CSS	33
HTML kiedyś i teraz	33
HTML kiedyś — aby dobrze wyglądało	33
HTML teraz — szkielet dla CSS	35
Pisanie HTML-a z myślą o CSS	35
Pamiętaj o strukturze	36
Dwa nowe znaczniki HTML do nauczania	36
Pamiętaj o układzie strony	40
O czym trzeba zapomnieć	41
Podstawowe wskazówki	43
Znaczenie deklaracji typu dokumentu	45
Aktualizowanie Internet Explorera	47
Rozdział 2. Tworzenie stylów i arkuszy stylów	49
Anatomia stylu	49
Zrozumieć arkusze stylów	52
Styl wewnętrzny czy zewnętrzny	52
Wewnętrzne arkusze stylów	53
Style zewnętrzne	54
Dołączanie arkusza stylów przy użyciu znacznika HTML	55
Dołączanie arkuszy stylów za pomocą kodu CSS	56
Kurs: tworzenie pierwszego stylu	57
Tworzenie stylu śródliniowego	57
Tworzenie wewnętrznych arkuszy stylów	58
Tworzenie zewnętrznego arkusza stylów	60

Rozdział 3. Selektory, czyli do czego odnoszą się style	67
Selektory znaczników — style dla całej strony	67
Selektor klasy — precyzyjna kontrola	69
Selektor ID — unikalne elementy strony	72
Nadawanie stylów grupom znaczników	72
Grupowanie selektorów	73
Selektor uniwersalny	74
Stylizowanie znaczników zagnieżdżonych	74
Drzewo rodzinne HTML	75
Tworzenie selektorów potomka	76
Tworzenie modułów	78
Pseudoklasy i pseudoelementy	79
Style odnośników	79
Stylizowanie fragmentów akapitu	80
Więcej pseudoklas i pseudoelementów	80
Selektory atrybutu	83
Selektor dziecka	85
Selektory dziecka z uwzględnieniem typu elementu	88
Selektor brata	89
Selektor :not()	90
Kurs: selektory	92
Tworzenie selektora grupowego	94
Tworzenie i stosowanie selektora klasy	95
Tworzenie selektora potomka	97
Tworzenie i stosowanie selektora identyfikatora	98
Ostatni szlif	100
 Rozdział 4. Oszczędzanie czasu dzięki dziedziczeniu	 103
Czym jest dziedziczenie?	103
Jak dziedziczenie upraszcza arkusze stylów	104
Granice dziedziczenia	104
Kurs: dziedziczenie	107
Prosty przykład: dziedziczenie jednopoziomowe	107
Wykorzystanie dziedziczenia do zmiany stylu całej strony	109
Kiedy dziedziczenie nie działa	111
 Rozdział 5. Zarządzanie wieloma stylami — kaskada	 113
Kaskadowość stylów	114
Style dziedziczone mogą się kumulować	114
Najbliższy przodek bierze górę	115
Górę bierze styl bezpośredni	116
Jeden znacznik, wiele stylów	116
Precyzja: który styl weźmie górę	118
Remis: wygrywa ostatni	119

Kontrolowanie kaskady	120
Zmienianie precyzji	121
Wybiórcze przesłanianie	122
Jak uniknąć wojny na precyzję	123
Resetowanie stylów	125
Kurs: kaskadowość w akcji	127
Resetowanie stylów i definiowanie nowych	127
Tworzenie stylu mieszanego	129
Rozwiązywanie konfliktów	130

Cześć II. Stosowanie CSS 133

Rozdział 6. Formatowanie tekstu 135

Czcionki	135
Wybór czcionki	137
Stosowanie czcionek sieciowych	140
Typy plików fontów	141
Kwestie prawne dotyczące czcionek	142
Gdzie szukać czcionek sieciowych	142
Generowanie różnych formatów czcionek	143
Dyrektywa @font-face	145
Tworzenie stylów przy użyciu czcionek sieciowych	147
Używanie wariantu pogrubionego i kursywy	148
Usługa Google Fonts	153
Znajdowanie i wybieranie czcionek	154
Korzystanie z usługi Google Fonts	157
Kolorowanie tekstu	159
Notacja szesnastkowa	160
Systemy HSL i HSLA	162
Zmiana rozmiaru pisma	163
Stosowanie pikseli	163
Stosowanie słów kluczowych, procentów i jednostki em	163
Formatowanie słów i liter	167
Pogrubienie i kursywa	167
Zamiana tekstu na wielkie litery	168
Dekorowanie tekstu	169
Odstęp między wyrazami i literami	170
Dodawanie cieni do tekstu	171
Formatowanie całych akapitów	172
Zmienianie odstępu między wierszami	172
Wyrównywanie tekstu	174
Wcinanie pierwszego wiersza i usuwanie marginesów	174
Formatowanie pierwszej litery lub pierwszego wiersza akapitu	177
Stylizowanie list	178
Typy list	178
Pozycjonowanie punktorów i numerów	179
Punktory graficzne	181

Kurs: formatowanie tekstu	181
Ustawienia strony	182
Formatowanie nagłówków i akapitów	185
Formatowanie list	188
Dostrajanie za pomocą klas	189
Wykańczanie projektu	191
Rozdział 7. Marginesy, dopełnienie i obramowanie	195
Istota modelu polowego	195
Marginesy i dopełnienie	197
Zapis skrótowy marginesów i dopełnienia	198
Konflikty marginesów	199
Likwidowanie odstępu za pomocą marginesów ujemnych	200
Elementy śródliniowe, blokowe i inne	202
Obramowanie	203
Skrócony zapis właściwości obramowania	204
Formatowanie poszczególnych krawędzi	205
Kolorowanie tła	206
Zaokrąglanie rogów	207
Cienie elementów	210
Określanie wysokości i szerokości	212
Obliczanie rzeczywistych wymiarów pól	213
Przedefiniowywanie wymiarów pól	214
Kontrolowanie wycieków za pomocą własności overflow	216
Określanie minimalnej szerokości i wysokości	218
Elementy pływające	219
Tła i obramowanie a elementy pływające	221
Pływaków nie wpuszczamy	222
Kurs: marginesy, tła i obramowanie	224
Ustawianie tła i marginesów	224
Ustawianie odstępów wokół znaczników	227
Tworzenie paska bocznego	230
O krok dalej	233
Rozdział 8. Umieszczanie grafiki na stronach WWW	235
CSS i znacznik 	235
Obrazy tła	236
Kontrola sposobu powtarzania obrazu w tle	239
Pozycjonowanie obrazu tła	242
Słowa kluczowe	242
Dokładne wartości	244
Procenty	244
Ustalanie położenia obrazu na sztywno	246
Definiowanie początku i końca tła	246
Skalowanie obrazów tła	248
Własność zbiorcza background	249
Ustawianie wielu obrazów w tle jednego elementu	251

Stosowanie gradientów w tle	254
Gradients liniowe	254
Powtarzanie gradientów liniowych	258
Gradients promieniste	260
Powtarzalne gradienty promieniste	262
Tworzenie gradientów przy użyciu narzędzia ColorZilla	262
Kurs: uatrakcyjnianie grafik	265
Definiowanie obramowania obrazu	265
Tworzenie galerii fotografii	268
Dodawanie cieni	271
Kurs: używanie obrazów tła	273
Umieszczanie obrazu w tle strony	273
Zastępowanie obramowania grafiką	275
Używanie grafiki w listach punktowanych	276
Uatrakcyjnianie paska bocznego	278
Rozdział 9. Upiększanie systemu nawigacji	281
Wybieranie odnośników do stylizacji	281
Poznaj stany odnośników	281
Wybieranie określonych odnośników	283
Stylizowanie odnośników	284
Podkreślanie odnośników	285
Tworzenie przycisku	286
Używanie grafiki	289
Tworzenie pasków nawigacji	290
Używanie list nienumerowanych	291
Pionowe paski nawigacji	292
Poziome paski nawigacji	293
Wczytywanie grafik tła odnośników z wyprzedzeniem	299
Stylizowanie wybranych rodzajów odnośników	301
Odnosiniki do innych witryn	301
Odnosiniki do adresów e-mail	302
Łączy do różnych typów plików	302
Kurs: stylizowanie odnośników	303
Podstawy formatowania odnośników	303
Dodawanie obrazu tła do odnośnika	305
Wyróżnianie różnych rodzajów odnośników	306
Kurs: tworzenie paska nawigacji	308
Dodawanie efektu rollover i tworzenie odnośników „Jesteś tutaj”	311
Z pionowego w poziomy	314
Rozdział 10. Przekształcenia, przejścia i animacje CSS	317
Przekształcenia	317
Obracanie	318
Skalowanie	319
Przesuwanie	321
Zniekształcanie	322

Stosowanie wielu przekształceń naraz	324
Punkt początkowy przekształcenia	324
Przejścia	326
Tworzenie przejść	327
Kontrola czasu wykonywania animacji	328
Opóźnianie początku przejścia	331
Własność zbiorcza przejść	332
Animacje	333
Definiowanie klatek kluczowych	334
Przypisywanie animacji do elementów	337
Kontrola przebiegu animacji	339
Kończenie animacji	340
Własność zbiorcza do definiowania animacji	342
Wstrzymywanie wykonywania animacji	343
Animacje i pseudoklasa :hover	344
Kurs	344
Dodawanie animacji	346
Rozdział 11. Formatowanie tabel i formularzy	351
Właściwy sposób używania tabel	351
Stylizowanie tabel	353
Dodawanie dopełnienia	354
Ustawianie wyrównania w pionie i w poziomie	354
Tworzenie obramowania	356
Stylizowanie wierszy i kolumn	357
Stylizowanie formularzy	359
Elementy HTML formularzy	361
Rozmieszczanie elementów formularza za pomocą CSS	362
Kurs: stylizowanie tabeli	364
Kurs: stylizowanie formularza	369
Cześć III. Tworzenie układu strony za pomocą CSS	375
Rozdział 12. Wprowadzenie do układów stron	377
Typy układów stron WWW	377
Jak działa układ w CSS?	380
Znacznik <div>	380
Elementy sekcyjne HTML5	382
Techniki rozmieszczania elementów na stronie	383
Strategie planowania układu	384
Zacznij od treści	384
Przede wszystkim mobilność	384
Rozpocznij od nakreślenia szkicu	385
Zidentyfikuj pola treści	386
Płyn z prądem	387
Pamiętaj o obrazach w tle	387

Fragmenty układanki	388
Rozmieszczanie elementów warstwowo	388
Pamiętaj o marginesach i dopełnieniu	388
Rozdział 13. Tworzenie układów opartych na elementach pływających	389
Stosowanie elementów pływających w układach	392
Używanie właściwości float dla wszystkich kolumn	393
Elementy pływające wewnątrz elementów pływających	395
Rozwiązywanie problemów z elementami pływającymi	395
Czyszczenie i obejmowanie elementów pływających	397
Tworzenie kolumn o pełnej wysokości	401
Zapobieganie upadaniu elementów pływających	407
Zastosowanie własności box-sizing	409
Kurs: układy wielokolumnowe	410
Strukturyzowanie HTML-a	411
Tworzenie stylów układu	412
Dodawanie kolejnej kolumny	413
Dodawanie wolnej przestrzeni	415
Ustawianie stałej szerokości	417
Mieszanie układu płynnego ze stałym	418
Rozdział 14. Projektowanie elastycznych witryn (RWD)	423
Podstawy techniki RWD	423
Przystosowywanie strony do techniki RWD	425
Zapytania o media	427
Strategie użycia zapytań o media	427
Definiowanie stopni układu	429
Od czego zacząć	430
Tworzenie zapytań o media	431
Zapytania o media wewnątrz arkuszy stylów	432
Podstawowa struktura arkusza stylów	433
Najpierw urządzenia przenośne	434
Elastyczne siatki	434
Jak ważna jest kolejność elementów w kodzie HTML	436
Resetowanie modelu polowego	436
Zamienianie układu sztywnego w elastyczną siatkę	437
Płynne obrazy	439
Wady stosowania płynnych obrazów	441
Filmy i animacje Flash	442
Kurs stosowania techniki RWD	443
Zmiana kolejności elementów HTML	443
Płynne obrazy	446
Definiowanie stylów dla tabletów	448
Definiowanie stylów dla telefonów	450

Rozdział 15. Pozycjonowanie elementów na stronie WWW	455
Jak działają właściwości pozycjonujące?	456
Ustawianie wartości pozycjonujących	458
Gdy pozycjonowanie bezwzględne jest względne	461
Kiedy (i gdzie) używać pozycjonowania względnego?	462
Stos elementów	464
Ukrywanie fragmentów strony	466
Użyteczne strategie pozycjonowania	469
Pozycjonowanie wewnątrz elementu	469
Wyłamywanie elementu poza blok	470
Użycie stałego pozycjonowania do tworzenia ramek za pomocą stylów CSS ...	471
Kurs: pozycjonowanie elementów strony	473
Wzbogacanie banera strony	474
Dodawanie podpisu do zdjęcia	477
 Cześć IV. Zaawansowany CSS	481
 Rozdział 16. CSS dla strony przeznaczonej do wydruku	483
Jak działają arkusze stylów dla mediów?	483
Jak dodawać arkusze stylów przeznaczone dla mediów?	485
Określanie typu medium dla zewnętrznego arkusza stylów	486
Określanie typu medium w arkuszu stylów	486
Tworzenie stylów dla wydruku	487
Używanie deklaracji !important do przesłonięcia stylów ekranowych	488
Zmiana stylów tekstu	488
Stylizowanie tła dla wydruków	490
Ukrywanie niepotrzebnych obszarów strony	491
Wstawianie podziałów stron w wydrukach	493
Kurs: tworzenie arkusza stylów przeznaczonego dla wydruków	494
Usuwanie niepotrzebnych elementów strony	494
Dostosowywanie układu	496
Zmiana formatowania tekstu	498
Wyświetlanie URL	499
 Rozdział 17. Dobre nawyki w CSS	501
Wstawianie komentarzy	501
Porządkowanie stylów i arkuszy stylów	502
Jasno nazywaj style	503
Używanie kilku klas dla zaoszczędzenia czasu	504
Grupuj style, aby utrzymać porządek w plikach	506
Korzystanie z wielu arkuszy stylów	507
Usuwanie przeszkadzających stylów przeglądarki	509
Wykorzystanie selektorów potomka	513
Dzielenie stron na sekcje	514
Zidentyfikuj treść właściwą dokumentu	515
Dostarczanie kodu CSS tylko dla Internet Explorera	518
Komentarze warunkowe dla różnych wersji Internet Explorera	520

Dodatki 521

Dodatek A. Zestawienie właściwości CSS 523

Wartości CSS	523
Kolory	523
Długości i rozmiary	525
Słowa kluczowe	526
Adresy URL	527
Właściwości tekstu	527
color (dziedziczona)	528
font (dziedziczona)	528
font-family (dziedziczona)	528
font-size (dziedziczona)	529
font-style (dziedziczona)	529
font-variant (dziedziczona)	529
font-weight (dziedziczona)	530
letter-spacing (dziedziczona)	530
line-height (dziedziczona)	530
text-align (dziedziczona)	530
text-decoration	530
text-indent (dziedziczona)	531
text-shadow (dziedziczona)	531
text-transform (dziedziczona)	531
vertical-align	532
white-space	532
word-spacing (dziedziczona)	532
Właściwości list	533
list-style (dziedziczona)	533
list-style-image (dziedziczona)	533
list-style-position (dziedziczona)	533
list-style-type (dziedziczona)	534
Dopełnienie, obramowania i marginesy	534
box-shadow	534
border	534
border-radius	535
border-top, border-right, border-bottom, border-left	535
border-color	535
border-top-color, border-right-color, border-bottom-color, border-left-color	536
border-style	536
border-top-style, border-right-style, border-bottom-style, border-left-style	536
border-width	537
border-top-width, border-right-width, border-bottom-width, border-left-width	537
box-sizing	537
outline-color	538
outline-style	538

outline-width	538
padding	538
padding-top	539
padding-right	539
padding-bottom	539
padding-left	539
margin	539
margin-top	540
margin-right	540
margin-bottom	540
margin-left	540
Tła	540
background	541
background-attachment	541
background-clip	541
background-color	542
background-image	542
background-origin	542
background-position	543
background-repeat	543
background-size	544
Właściwości układu strony	544
bottom	544
clear	544
clip	545
display	545
float	546
height	546
left	547
max-height	547
max-width	547
min-height	548
min-width	548
overflow	548
position	549
right	549
top	549
visibility	550
width	550
z-index	551
Własności animacji, przekształceń i przejść	551
@keyframes	551
animation	552
animation-name	552
animation-duration	553
animation-timing-function	553
animation-delay	553
animation-iteration-count	553
animation-direction	554

animation-fill-mode	554
animation-play-state	554
transform	554
transform-origin	555
transition	555
transition-property	555
transition-duration	556
transition-timing-function	556
transition-delay	556
Właściwości tabel	556
border-collapse	557
border-spacing	557
caption-side	557
empty-cells	558
table-layout	558
Pozostałe właściwości	558
content	558
cursor	559
opacity	559
orphans („sieroty”)	560
page-break-after	560
page-break-before	560
page-break-inside	561
widows	561
Dodatek B. Zasoby CSS	563
Podręczniki	563
World Wide Web Consortium (W3C)	563
Książki	564
Inne źródła online	564
Pomoc dotycząca CSS	564
Fora dyskusyjne	564
Porady, sztuczki i wskazówki	565
Nawigacja z CSS	565
Kursy	565
Przykłady	566
Układy oparte na CSS	566
Informacje o modelu polowym	566
Przykłady układów	567
Inne zasoby z układami	567
Witryny pokazowe	568
Książki na temat CSS	568
Edytory CSS	568
Windows i Macintosh	569
Tylko Windows	569
Tylko Macintosh	569
Skorowidz	571

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.

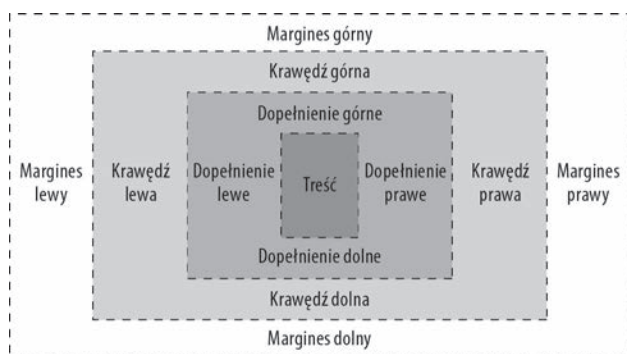
Wszystkie właściwości opisane w tym rozdziale wzięte razem składają się na jedną z najważniejszych koncepcji w CSS — model polowy (zwany też *modelem blokowym* lub *pudełkowym*).

Istota modelu polowego

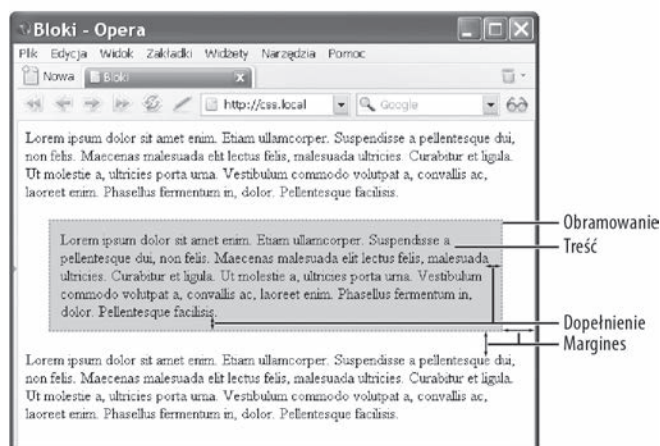
Mówiąc o akapicie lub nagłówku, zawsze mamy na myśli litery, słowa i zdania. Prawdopodobnie więc mówiąc o znaczniku `<img />`, mamy na myśli zdjęcie, logo albo jeszcze jakiś inny rodzaj obrazu. Przeglądarka jednak traktuje je (i wszystkie inne znaczniki) inaczej — jako małe pola. Dla przeglądarki każdy 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



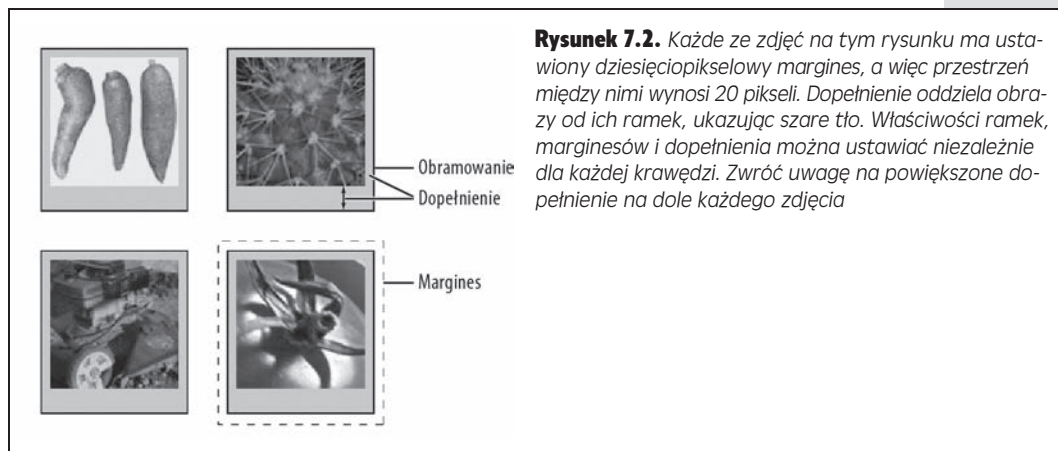
- **Kolor tła** (background-color) wypełnia powierzchnię wewnątrz ramek, łącznie z obszarem zajmowanym przez dopełnienie.
- **Margines** (margin) to przestrzeń oddzielająca od siebie poszczególne znaczniki. Na przykład marginesem jest powszechnie spotykany odstęp między górną a dolną krawędzią akapitów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



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).

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 dziesięcioprocentowy 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; 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;
}
```

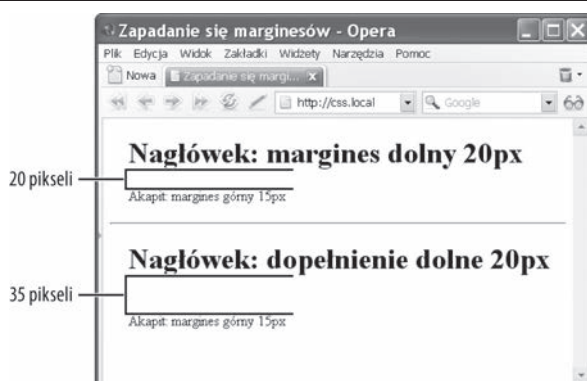
W podobny sposób tworzy się jednakowy odstęp pomiędzy treścią a obramowaniem ze wszystkich stron elementu:

```
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 marginesy 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).

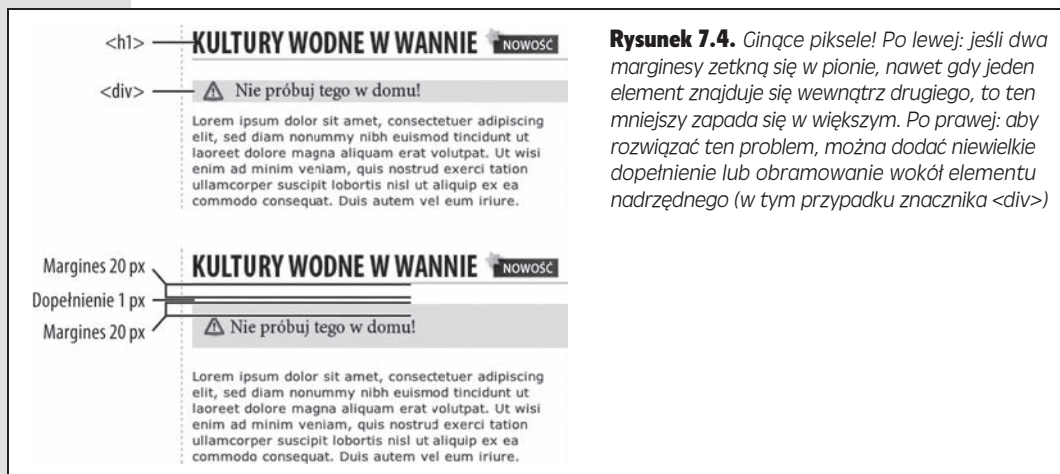


Rysunek 7.3. Gdy stykają się dwa marginesy w pionie, to mniejszy z nich zapada się w tym większym. Mimo że górny nagłówek ma margines dolny o szerokości 20 pikseli, a akapit górny o szerokości 15 pikseli, przeglądarka utworzy tylko dwudziestopikselowy odstęp między nimi. Aby uzyskać 35 pikseli odstępu, 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 `<div>`). 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 nagłówki, 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 `<div>`. W tym przypadku nie ma znaczenia rozmiar marginesu zastosowanego do nagłówka — i tak zawsze będzie znajdował się nad znacznikiem `<div>`.



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 `<div>` lub obramowanie. Jako że obramowanie i dopełnienie znajdują się *między* dwoma marginesami, zapobiegają ich stykaniu się, dzięki czemu nagłówek zyskuje nieco przestrzeni (rysunek 7.4, po prawej).

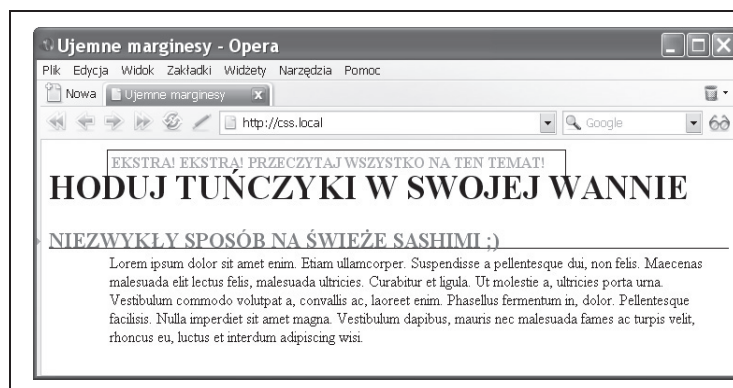
Uwaga: Marginesy stykające się w poziomie (lewy i prawy) oraz pomiędzy elementami pływającymi nie zapadają się, podobnie jak marginesy elementów pozycjonowanych względnie i bezwzględnie, o których będziemy mówić w rozdziale 15.

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!” dotyczą 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.

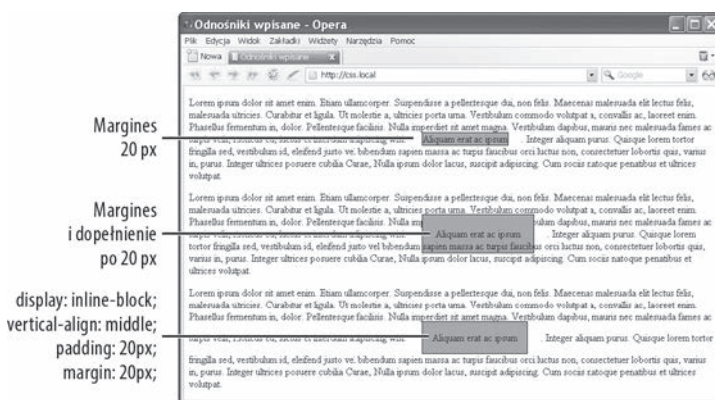
Czasami możemy zechcieć, aby element śródliniowy zachowywał się jak blokowy i odwrotnie. Elementy list wypunktowanych są oddzielnymi elementami blokowymi — ustawione są jeden nad drugim. Ale co zrobić, jeśli chcemy to zmienić w taki sposób, aby elementy listy znajdowały się obok siebie w jednej linii (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 `tlo` i obramowanie odnośnika nakładają się na tekst znajdujący się nad i pod nim. `Tlo` 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 `<strong>`, `<em>` i `<span>`. Należy też dodać deklarację `vertical-align: middle`, aby przeglądarka IE 7 wyświetlała element śródliniowo-blokowy tak samo jak inne przeglądarki

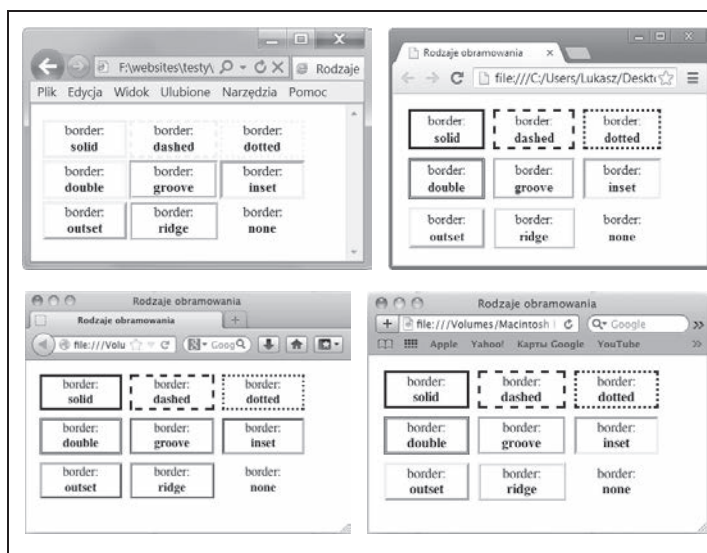
Uwaga: Właściwość `display` może przyjmować mnóstwo wartości, z których część nie jest jeszcze obsługiwana przez wszystkie przeglądarki. Wartość `inline-block` rozpoznają wszystkie aktualnie używane przeglądarki (rysunek 7.6). Jest jednak wartość `none`, którą obsługują prawie wszystkie przeglądarki 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 punkt. 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 kresiek. 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 jakimkolwiek 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 prześwitywać znajdujące się pod spodem kolory, tekstury i obrazy. Można na przykład dla całej strony ustawić jasnobłękitne 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

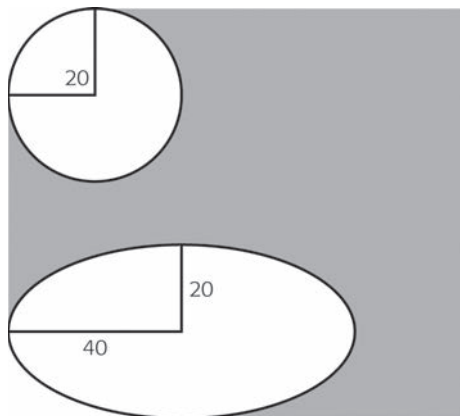
Wcześniej napisałem, że dla przeglądarek każdy element jest prostokątem, o czym można się łatwo przekonać, definiując dowolnemu elementowi obramowanie. Można jednak te kanty ładnie zaokrąglić, aby wyglądały tak, jak na przykład na rysunku 7.8. W CSS3 wprowadzono nową własność o nazwie `border-radius`, 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 (choć 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 Firefoxie 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.

```
box-shadow: 4px 6px 8px rgba(0,0,0,.75);
```

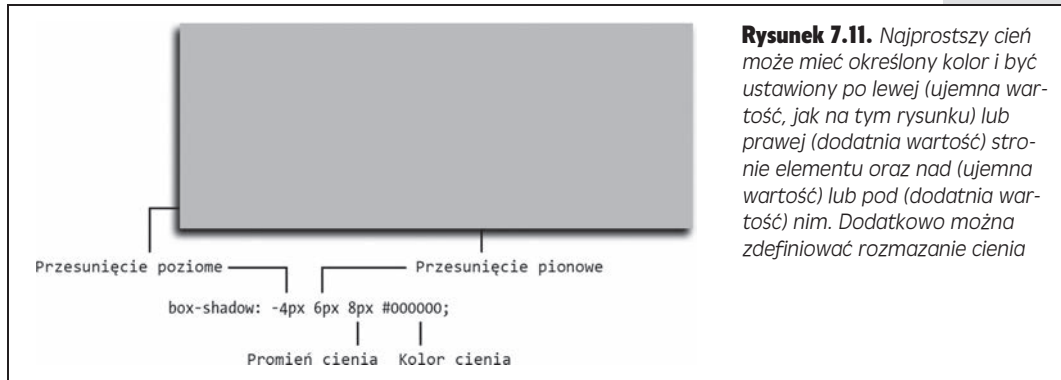
```
box-shadow: 0 0 8px 2px rgba(0,0,0,.75);
```

```
box-shadow: -4px -6px 8px 2px rgba(0,0,0,.5),  
            8px 6px 12px rgba(0,0,0,.5);
```

```
box-shadow: inset 0 0 8px 2px rgba(0,0,0,.75);
```

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.



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 krawędź elementu (jak na gorze na rysunku 7.10), a ujemna — nad górną.

Uwaga: Do definiowania cieni elementów należy używać jednostek em lub pikseli. Procenty są nie-
dozwolone.

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 wpisujemy `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 miniatur 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;
```

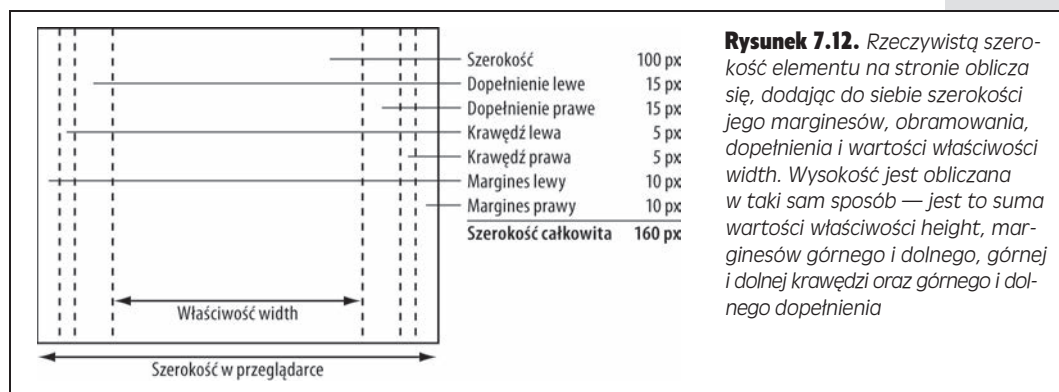
Piksele to... piksele. Są łatwe do opanowania i nie sprawiają problemów w zastosowaniu. Tworzą też dokładne wymiary, które się nie zmieniają. Jednostka *em* odpowiada rozmiarowi tekstu w stylizowanym elemencie. Załóżmy, że ustawiliśmy rozmiar pisma na 24 piksele, jeśli więc szerokość ustawimy na 2 em,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



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 $50\% + \text{suma szerokości lewego i prawego dopełnienia} + \text{suma szerokości lewej i prawej krawędzi obramowania}$). 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 założmy, 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 ($100 + 20 [\text{lewe dopełnienie}] - 20 [\text{prawe dopełnienie}] = 60$).
- 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 niezwykle 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).

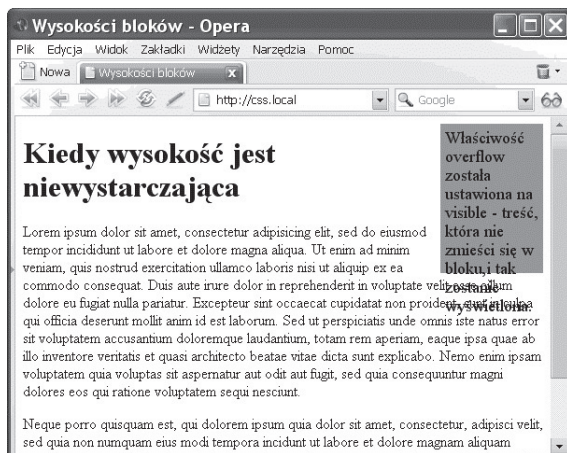
Uwaga: Własność `box-sizing` działa nawet w przeglądarce Internet Explorer 8, a więc jest obsługiwana przez jakieś 95% wszystkich obecnie używanych przeglądarek. Niestety, nie rozpoznaje jej Internet Explorer 7, przez co w tej przeglądarce elementy z ustawieniem `border-box` mogą być znacznie szersze,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

Kontrolowanie wycieków za pomocą własności `overflow`

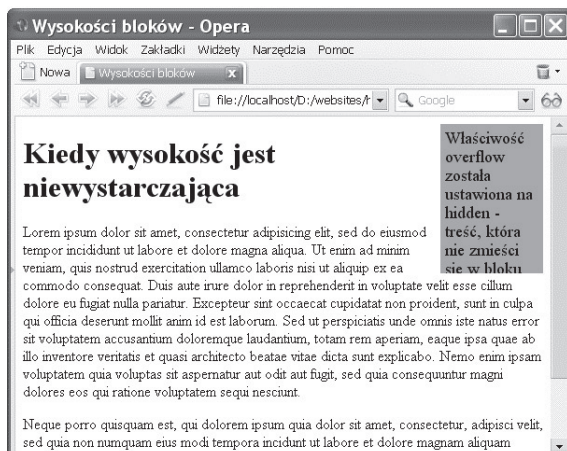
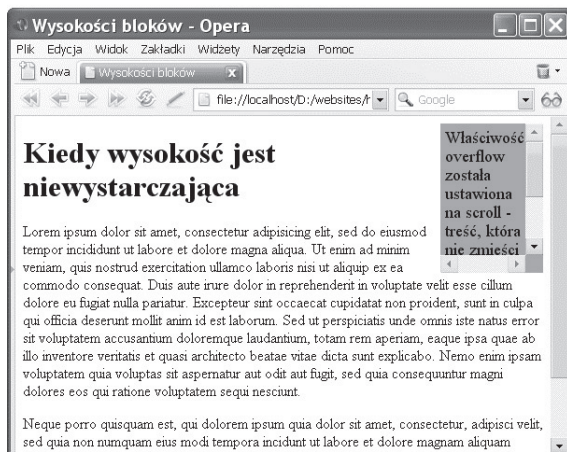
Jeśli treść elementu jest większa niż zdefiniowane w stylu wysokość i szerokość, to mogą mieć miejsce pewne nieprzewidywalne zdarzenia. Jak pokazano na rysunku 7.13, przeglądarki pozwalają treści wylać się na zewnątrz (za obramowanie i często na inną treść).

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:

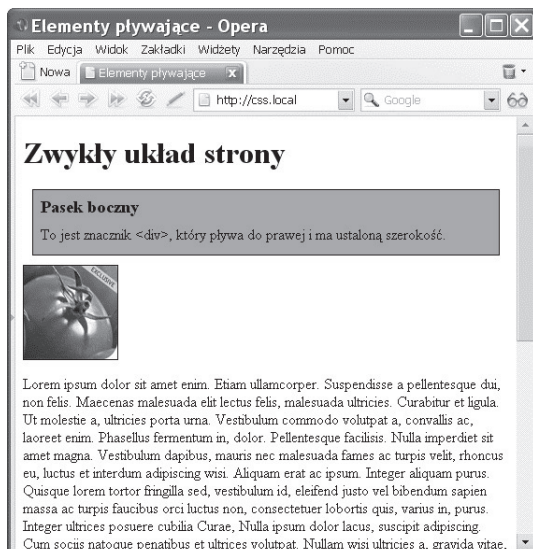
```
float: left;
```

- **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.

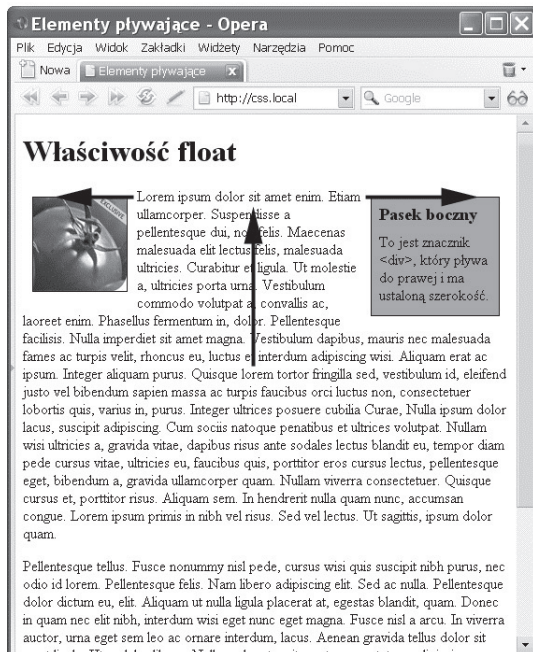
Elementy pływające dryfują do lewej lub prawej krawędzi okna przeglądarki albo *zawierającego je elementu*. Czasami tym zawierającym elementem jest okno przeglądarki, ale jeśli element pływający znajduje się w innym elemencie o ustalonej szerokości, to będzie przesunięty do lewej lub prawej krawędzi 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 `<div>` 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

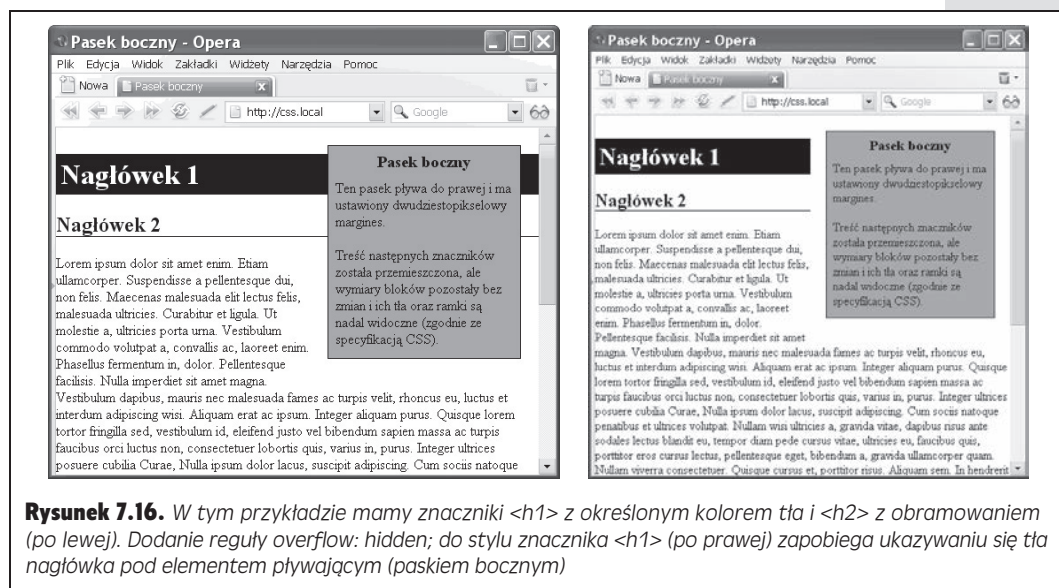


wyróżnione i inne elementy tworzące samodzielną całość (w kursie do tego rozdziału przedstawiłem tego przykład). Stosując właściwość `float` do elementów blokowych, dobrze jest też pamiętać o podaniu ich szerokości. W ten sposób można kontrolować ilość przestrzeni zajmowanej w poziomie przez blok oraz 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.



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.
- **right** — 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 owijała 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 obrazy 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.

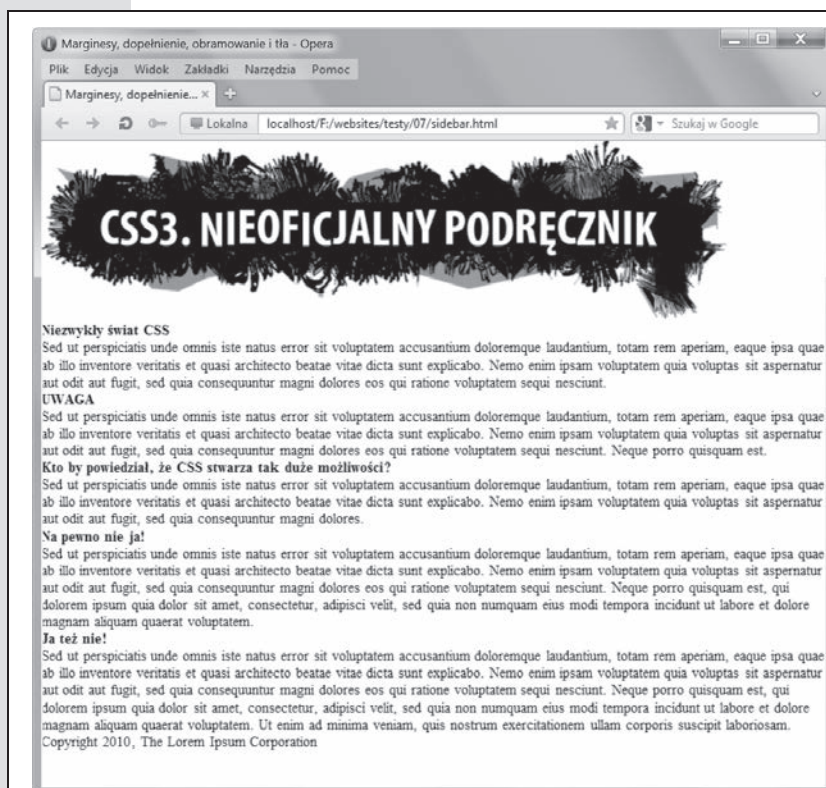
Kurs: marginesy, tła i obramowanie

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):

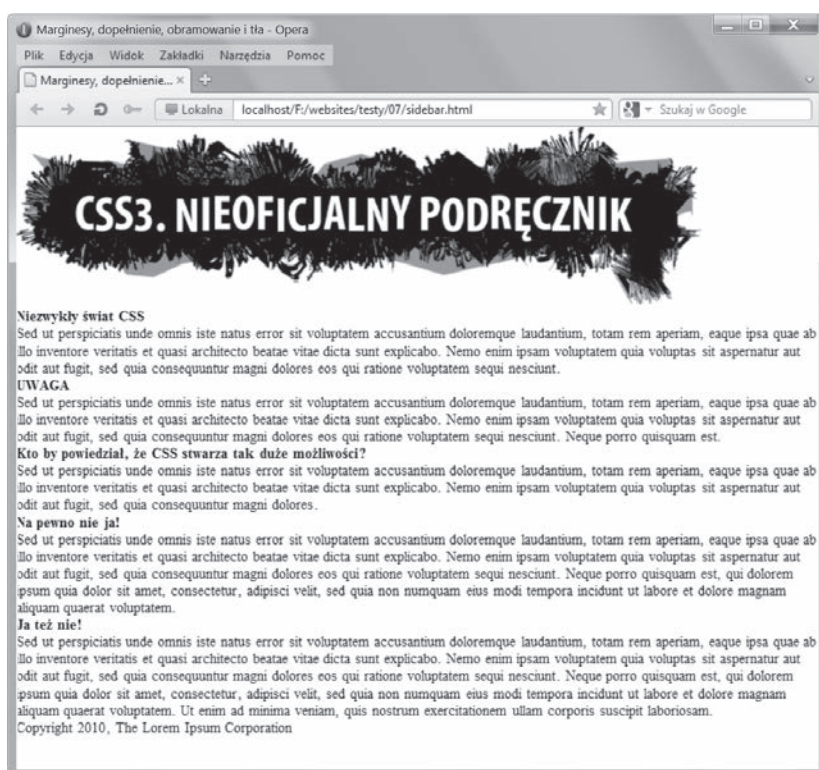
```
body {
    background-color: rgb(255,255,255);
    border: 3px solid rgb(75,75,75);
    box-shadow: 0 0 15px 5px rgba(44,82,100,.75);
    width: 760px;
    margin-top: 20px;
    margin-left: auto;
    margin-right: auto;
    padding: 15px;
}
```

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 wyrównanie. 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 wyrównanie 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 wyrównać 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 podejrzuj ją w przeglądarce. Dzięki temu lepiej zrozumiesz, w jaki sposób działają poszczególne własności CSS.

2. Dodaj kolejną (wyróżnioną) 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 zawierające go kontenera.

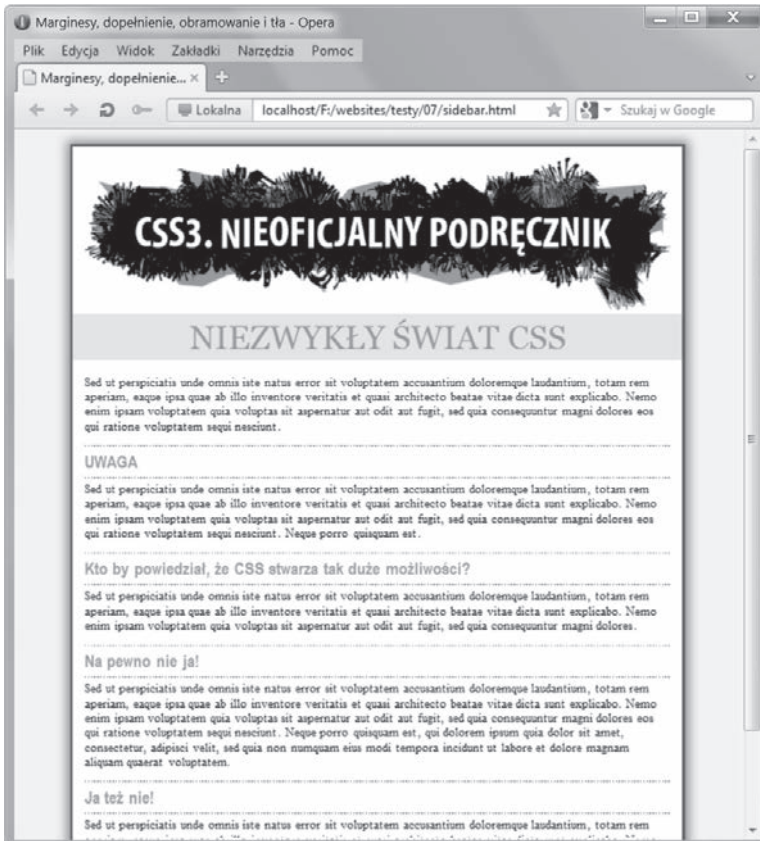
5. Teraz zdefiniujemy formatowanie elementów `h2`. Wpisz poniższą regułę pod regułą `h1`:

```
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. **Przełącz się 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 stylów 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 regułą 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.

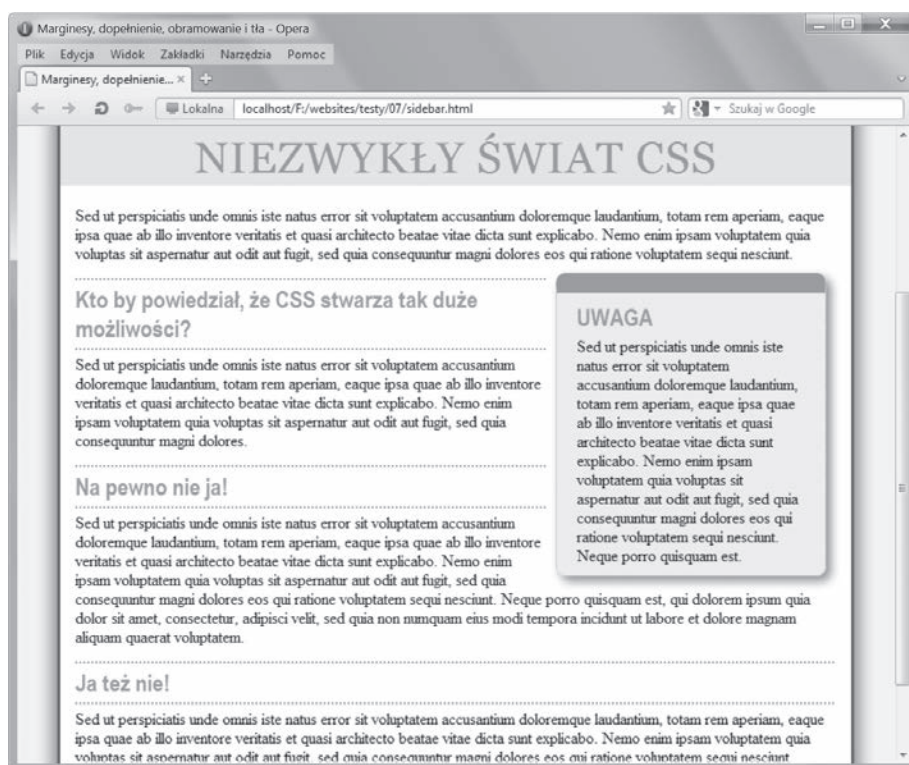
9. Odszukaj regułą z selektorem `h2` i dodaj do niej deklarację własności `overflow`:

```
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

0 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 (pamiętaj: użyj właściwości `text-transform` opisanej w rozdziale 6.). Po utworzeniu stylu zastosuj go do akapitu z notką o prawach autorskich.

Skorowidz

A

- adres URL, 23, 240, 527
- animacja, 326–332
- animacja koloru tła, 335
- animacje
 - CSS3, 333–344
 - Flash, 442
- animowane podpisy, 480
- arkusz stylów
 - dla mediów, 483
 - dla wydruków, 494
- arkusze stylów
 - wewnętrzne, 52, 58, 93
 - zewnętrzne, 54, 60, 93
- atrybut
 - class, 70, 505
 - href, 56
 - src, 85
 - title, 83
- atrybuty selektorów, 83

B

- blok deklaracji, 51
- bloki, 510

C

- cienie
 - elementów, 210, 211
 - pod zdjęciami, 271

- CSS, Cascading Style Sheets, 19
- CSS 2.1, 27
- CSS dla IE, 518
- CSS sprites, 299
- CSS3, 27
- CSS-Positioning, 455
- cudzysłowy, 512
- czas trwania animacji, 328, 339, 348
- czcionki, 135
 - awaryjne, 148
 - bezseryfowe, 138
 - darmowe, 143
 - nieproporcjonalne, 139
 - plakatowe, 155
 - sieciowe, 140, 142, 147
 - seryfowe, 138
 - w IE 8, 151, 153
- czyszczenie elementów pływających, 397

D

- darmowe
 - czcionki sieciowe, 143
 - kolekcje obrazów, 251
- definiowanie
 - czcionek, 140
 - klatek kluczowych, 334
 - obramowania obrazu, 265
 - przejścia, 326
 - wariantów czcionek, 151
 - zapytań o media, 432

deklaracja, 51
 !important, 121, 488, 499
 display
 block, 510
 inline-block, 294
 none, 467
 overflow
 hidden, 315, 399, 497
 typu dokumentu, 21, 45
 vertical-align, 370
dekorowanie tekstu, 169
długość, 525
dodawanie
 animacji, 346
 cieni, 171, 271
 dopełnienia, 354
 efektu rollover, 311
 elementu czyszczącego, 399
 klas do elementów, 513
 klatek, 347
 kolumny, 413
 obrazu tła, 305
 podpisu, 477
 ramek, 224
 wariantów czcionek, 149
 wolnej przestrzeni, 415
 zapytań o media, 432
dołączanie arkusza stylów, 55, 123
dopełnienie, 195–198, 236, 354,
 388, 534
dostosowanie
 wyglądu strony do urządzeń,
 427–429
 układu, 496
drukowanie, 487
drzewo znaczników HTML, 75, 77
dyrektywa
 !important, 121, 488, 499
 @font-face, 145–149, 182
 @import, 55, 120, 432, 508
 @media, 433, 486
dziedziczenie, 74, 103–120, 128
dziedziczenie jednopoziomowe, 107
dzielenie stron na sekcje, 514

E

edytory CSS, 568
efekt rollover, 311, 312
efekty przejścia, 333

elastyczne siatki, 434–439
element, *Patrz* znacznik
elementy
 HTML, 22, 36
 HTML formularzy, 361
 HTML5, 38
 pływające, 216, 221, 296, 389–421
 pozycjonowane absolutnie, 410
 sekcyjne HTML5, 382
 stałe, 456
 śródliniowe, 202
 tła, 490
EOT, Embedded Open Type, 141

F

fałszywe kolumny, 405
filmy, 442
fonty, 141
fonty ikoniczne, 148
format, *Patrz także* pliki
 EOT, 141
 GIF, 238
 JPEG, 238
 OpenType, 141
 PNG, 238
 SVG, 141
 TrueType, 141
 WOFF, 141
formatowanie
 akapitów, 172, 185
 formularzy, 359
 kolumn tabeli, 87
 krawędzi, 205
 list, 188
 modułów kodu, 78
 nagłówków, 185
 odnośników, 303
 pierwszej litery, 177
 przycisku nawigacji, 516
 słów, 167
 tabel, 351
 tabeli, 365
 tekstu, 135, 176, 181
 wielu elementów, 508
 wybranych elementów, 513
formaty grafiki, 238

- formularze, 359
 - legenda, 361
 - menu rozwijane, 361
 - pola tekstowe, 361
 - pola wyboru, 361
 - przyciski, 361
 - przyciski opcji, 361
 - zestaw pól, 361
- funkcja
 - rotate, 318
 - scale, 319, 320
 - skew, 323
 - translate, 321, 322

G

- galeria fotografii, 268
- generator układu, 394
- generyczny krój czcionki, 137
- gęstość pikseli, 164
- gradient
 - liniowy, 254, 257, 405
 - promienisty, 260
 - tła, 368
- grupowanie
 - odnośników, 284
 - selektorów, 73
 - stylów, 506

H

- heksadecymalny zapis koloru, 160
- HSL, hue, saturation, lightness, 162
- HTML, Hypertext Markup Language, 21
- HTML5, 24

I

- identyfikatory
 - w CSS, 72
 - w JavaScript, 73
- identyfikowanie stylów, 502
- Internet Explorer 8, 39, 46, 151

J

- JavaScript, 329
- jednostka rozmiaru tekstu, 163
 - em, 165
 - piksel, 163

- procent, 165
- rem, 167, 212
- słowa kluczowe, 164
- język HTML, 20

K

- kanał alfa, 161
- kapitaliki, 169
- kaskada, cascade, 113, 120, 128
- kaskadowość, 127
- klasa

- .announcement, 340
- .break_after, 493
- .break_before, 493
- .circle, 178
- .clear, 410
- .fade, 341
- .footer, 496
- .mainWrapper, 496
- .sidebar, 231, 232
- footerMain, 497
- hat, 467
- homeLink, 313
- main, 129
- navButton, 327
- pageStyle, 110
- storyNav, 515

- klatka kluczowa, keyframe, 333

- klikanie, 28

- kolejność

- elementów, 436, 443
- elementów potomnych, 465
- znaczników, 221
- znaczników <div>, 394

- kolor tła, 196, 206, 275

- kolorowanie tekstu, 159

- kolory, 523

- kolory SVG, 160

- kolumna tabeli, 358

- komentarze, 501, 507

- komentarze warunkowe, 519, 520

- konflikt

- marginesów, 199

- stylów, 113, 127

- właściwości, 130

- właściwości dziedziczonych, 115

- koniec wartości atrybutu, 85

- kontekst pozycjonowania, 463

kontener zbiorczy <div>, 387
kontrola przebiegu animacji, 328, 339
kończenie animacji, 340
krawędź obramowania, 357
krzywa Béziera, 330
kursywa, 148, 153, 167

L

linia pod odnośnikiem, 285
lista, 90, 178
 nienumerowana, 291
 numerowana, 34
 punktowana, 36, 276
lokalizacja
 sekcji strony, 73
 zewnętrznego pliku, 56

Ł

łącza wewnętrzne, 301
łącze hipertekstowe, 23
łączenie arkuszy stylów, 505

M

margines, 196, 236, 388, 534
 między akapitami, 175
 ujemny, 200
maszyna wirtualna, 48
menu, 29, 297
metoda Micro Clear Fix, 400
mieszanie układów, 418
minimalizacja arkuszy stylów, 505
mobilność, 384
model
 elastycznych pól, 406
 połowy, 196, 436
moduł
 elastycznego rozmieszczenia pól, 385
 układów siatkowych, 385, 406
 układów wielokolumnowych, 385, 401

N

nagłówek, head, 23
nagłówek tabeli, 353
najbliższy przodek, 115
nakładanie elementów, 475

nawigacja z CSS, 565
nazwa animacji, 338
nazwy stylów, 503
notacja szesnastkowa, 160
numeracja, 179

O

obiekty CSS, 518
oblewanie, floating, 236
obliczanie
 precyzji selektorów, 118
 wymiarów pól, 213–215
obracanie, 318
obramowanie, 195, 203, 235, 356, 534
 elementu pływającego, 222
 obrazu, 265
obraz tła, 236, 251, 273, 286, 387
obrazy, 251
obsługa
 obrazów w IE 8, 279
 przezroczystości w IE 8, 162
 stylizowania formularzy, 360
odkrywanie podpisu, 479
odnośnik
 Funkcje, 313
 Strona główna, 313
odnośniki, 281
 do adresów e-mail, 302
 do innych witryn, 301
 do plików, 302
 na wydrukach, 492
odstęp
 między akapitami, 175, 230
 między kolumnami, 417
 między komórkami, 356, 367
 między literami, 170
 między wierszami, 100, 172
 między wyrazami, 170
odstępy wokół znaczników, 227
ograniczenia selektora :not(), 91
określanie
 typu medium, 486
 wymiarów obiektu, 212
opcja static, 457
opóźnianie przejścia, 331
osadzanie
 czcionek sieciowych, 157
 zapytań o media, 433

P

pamięć podręczna przeglądarki, 53, 511

pasek

- boczny, 278, 389, 411

- nawigacji, 291, 297, 308

- przewijania, 216

piksel, 212, 442

planowanie układu, 384

plik

- another_page.html, 64

- at-font-face.css, 184

- banner.html, 344

- base.css, 508

- basic.html, 57, 62

- bg_images.html, 273, 278

- cascade.html, 127

- css3n3.htm, 224

- form.html, 369

- gallery.html, 268

- hats.html, 474, 477

- image.html, 265

- inheritance.html, 108

- links.html, 303

- main.txt, 411

- nav_bar.html, 308, 314

- new-source-order.html, 444

- print.css, 495, 500

- print.html, 494

- reset.css, 126, 183

- rwd.html, 443, 454

- selector_basics.html, 92, 95

- sidebar.html, 225

- sidebar1.txt, 411

- sidebar2.txt, 413

- start.html, 411, 417

- styles.css, 61

- table.html, 365

- text.html, 182, 191

pliki

- .css, 54

- .eot, 147

- .otf, 141

- .svg, 147

- .ttf, 141, 147

- .woff, 147

płynne obrazy, 439–441, 446

podkreślanie odnośników, 285

podział stron, 493

pogrubienie, 148, 153, 167

poła treści, 386

pole

- blokowe, block box, 202

- śródliniowe, inline box, 202

polecenie Podgląd wydruku, 488

porządkowanie stylów, 502

posteryzacja, 238

powiększanie

- przycisku, 344

- tekstu, 164

powtarzanie

- animacji, 341

- gradientów liniowych, 258

- gradientów promienistych, 262

- obrazu w tle, 239

pozycjonowanie

- bezwzględne, 456, 459, 476

- elementów, 455

- elementów strony, 473

- numerów, 179

- obrazu tła, 242–246

- punktorów, 179

- stałe, 471–473

- statyczne, 458

- tła elementu, 300

- wewnątrz elementu, 469

- względne, 456, 462, 467

- znaczników, 461

precyzja, specificity, 119, 121

precyzja selektorów, 118

preprocesory CSS, 505

program

- Coda2, 26

- Dreamweaver, 26

- EditPlus, 26

- Expression Web 2, 26

- Gridinator, 394

- InDesign, 383

- jEdit, 25

- Notepad++, 25

- Quark Xpress, 383

- skEdit, 26

- Sublime Text, 26

- TextWrangler, 25

- Ultimate CSS Gradient Generator,
262

- Webfont Generator, 143

- projekt
 - dwukolumnowy, 413
 - elastyczny, 379
 - wielokolumnowy, 419
 - projektowanie elastyczne, 378, 423–454
 - elastyczne media, 425
 - elastyczne siatki, 425
 - zapytania o media, 425
 - projekty nawigacji, 299
 - przedrostek
 - moz-, 209
 - ms-, 209
 - o-, 209
 - webkit-, 209
 - przeglądarki mobilne, 425
 - przejście, transition, 326 *Patrz także*
 - animacje CSS3
 - przekształcanie
 - listy w pasek nawigacji, 308
 - paska nawigacji, 315
 - pojemnika, 399
 - przekształcenia trójwymiarowe, 325
 - przesłanianie
 - stylów ekranowych, 488
 - wybiórcze, 122
 - przesuwanie, 321
 - przezroczystość, 161, 238
 - przycisk, 286
 - przycisk nawigacji, 516
 - przyciski formularza, 361
 - przykładowe pliki, 30
 - przypisywanie animacji, 337
 - pseudoelement, 79
 - :first-letter, 80
 - :first-line, 80
 - :selection, 82
 - after, 82
 - before, 81
 - first-line, 178
 - last-child, 86
 - pseudoklasa, 79
 - :hover, 290
 - :visited, 290
 - :active, 326
 - :target, 326
 - :focus, 326
 - checked, 364
 - enabled, 364
 - first-child, 86
 - first-of-type, 88
 - focus, 81, 282, 364, 373
 - hover, 282, 299, 305, 344
 - last-of-type, 88
 - nth-child, 86
 - nth-of-type, 88
 - visited, 282
 - pseudoklasy formularzy, 364
 - punkt początkowy przekształcenia, 324
 - punktacja, 179
 - punktory graficzne, 181
 - pusty wiersz, 129
- ## R
- reguła
 - .announcement, 338
 - .logo, 349
 - .main, 417, 445
 - .pageWrapper, 418
 - .sidebar, 128, 445
 - @keyframes, 338, 346–348
 - nav, 453
 - przycisku, 372
 - reguły CSS, 127
 - reset
 - CSS, 125, 225
 - Erica Meyera, 126
 - resetowanie
 - modelu polowego, 436
 - stylów, 125, 225
 - stylów przeglądarek, 512
 - RGB, red, green, blue, 160
 - RGBA, red, green, blue, alpha, 161
 - rodzaje odnośników, 301, 306
 - rozmiar, 525
 - czcionki, 489
 - marginesów, 176
 - obrazu tła, 249
 - piksele, 442
 - tekstu, 163
 - rozmieszczanie elementów, 406
 - formularza, 362
 - na stronie, 383
 - równomierne, 270
 - warstwowo, 388
 - rozsuwanie elementów listy, 189
 - rozwiązywanie konfliktów, 130
 - RWD, Responsive Web Design, 378, 423

S

scalanie marginesów, collapsing

margins, 200

selektor, 51, 67

::selection, 82

.button, 287

a:visited, 305

a.button, 287

first-child, 88

first-of-type, 88

focus, 81

not(), 90, 302

nth-child(), 89

nth-of-type(), 89, 357, 368

selektory

atrybutu, 83

brata, 89, 100

dziecka, 85, 88

grupowe, 94, 117, 313, 420, 495

identyfikatora, 72, 98

klas, 69, 95, 503

uniwersalne, 74

potomka, 74, 97, 284, 370, 513

złożone, 117

znaczników, 67, 74, 95

serwer TypeKit, 159

serwis

Google Web Fonts, 154

HTML5 Doctor, 39

siatka, 397

skalowanie, 319

obrazów tła, 248

szerokości, 321

wysokości, 321

składanie przekształceń, 324

skrótów klawiaturowe, 29

słowa kluczowe kolorów, 160

słowo kluczowe, 524

all, 327

alternate, 341

auto, 216

bottom, 243

center, 242

closest-corner, 261

closest-side, 260

contain, 249

cover, 249

even, 87

farthest-corner, 261

farthest-side, 261

hidden, 216

left, 219, 242

none, 219

odd, 87

right, 219, 242

rotate, 317

scroll, 216

top, 243

url, 181

visible, 216

sprawdzanie kodu

arkuszy stylów, 55

HTML, 42

sprite CSS, 300

stała szerokość, 378, 392, 417

stałe elementy, 456

stan odnośnika

:active, 80, 282

:hover, 79, 286

:link, 79, 285

:visited, 79, 282

stopnie

kolorów, color stop, 256

układu, 429

stopniowanie gradientu, 256

stos elementów, 464

stosowanie

CSS, 133

czcionek sieciowych, 159

elementów pływających, 296, 392

grafiki, 289

grafiki w listach, 276

klas do znacznika, 506

obrazów tła, 273

pozycjonowania stałego, 471

pozycjonowania względnego, 462

RWD, 443

tabel, 351

strategie pozycjonowania, 469

strona internetowa, 40

struktura arkusza stylów, 433, 434

styl, 49

.columnWrapper, 450

.gallery figcaption, 479

.headHighlight, 490

.intro, 131

.inventory td, 367

styl

- .main h2, 276
- .mainNav, 310
- .redhighlight, 503
- .sidebar2, 449
- header, 475
- nav a, 345

style

- bezpośrednie, 116
- całej witryny, 61
- dla tabletów, 448
- dla telefonów, 450
- dla wydruku, 487
- dziedziczone, 114
- marginesów, 111
- menu rozwijanego, 373
- mieszane, 129
- odnośników, 79, 283
- ramek, 204
- śródliniowe, 57
- tekstu, 488
- układu, 412

stylizowanie

- formularza, 369
- formularzy, 359, 362
- grupy elementów, 515
- list, 178
- odnośników, 284, 301, 303
- przycisku, 371
- tabel, 353
- tabeli, 364
- tła dla wydruków, 490
- wierszy i kolumn, 357

swobodny układ strony, 393

symbol

- #, 72
- *, 74

symulowanie trójwymiarowości, 323

system

- HSL, 162
- HSLA, 162
- RGB, 160
- RGBA, 161

szerokość

- elementów pływających, 407
- elementu, 213
- maksymalna, 218
- okien przeglądarki, 379

szkic struktury, 386

szkielet dokumentu, 21

Ś

ścieżka względna, 240

T

tabele, 351

technika RWD, *Patrz* projektowanie elastyczne

testowanie

- projektów RWD, 439
- stron, 48

tło, 540

- elementu <body>, 225
- nagłówka, 221
- strony, 206
- z wieloma obrazami, 251, 253

treść, body, 23, 384

tryb wstecznej zgodności, 45

tworzenie

- animacji, 333
- elastycznych obrazów, 448
- galerii fotografii, 268
- gradientów, 254, 262
- kolumn, 401
- list, 90
- menu, 296
- modułów, 78
- obramowania, 356
- odnośników, 311
- paska bocznego, 230
- paska nawigacji, 308
- pasków nawigacji, 290
- przejsć, 327
- przycisku, 286
- selektora grupowego, 94
- selektora identyfikatora, 98
- selektora klasy, 95
- selektora potomka, 97
- selektorów potomka, 76
- selektorów potomków, 517
- stylów, 49
- stylów dla wydruku, 487
- stylów układu, 412
- stylu klasy, 69, 71
- stylu mieszanego, 129
- stylu śródliniowego, 57
- szkicu struktury, 386

- tabel, 352
- układu strony, 375
- wewnętrznych arkuszy stylów, 58
- zapytań o media, 431
- zewnętrznego arkusza stylów, 60

typy

- adresów URL, 240
- dokumentu, 21
- list, 178
- łącz, 55
- mediów, 484–486
- plików fontów, 141
- układów stron, 377

U

ujemny margines, 192

układ strony, 40, 224, 375

- elastyczny, RWD, 378, 423
- o stałej szerokości, 378
- płynny, 378, 389
- sztwywny, 378, 437

układy

- dwukolumnowe, 391
- oparte na CSS, 566
- trójkolumnowe, 393
- wielokolumnowe, 410

ukrywanie

- elementów, 491
- fragmentów strony, 466
- podpisów, 479
- punktorów, 276
- treści, 216

ułatwienia dla CSS3, 505

umieszczanie obrazu w tle, 273

upadanie elementów pływających, 407

URL, Uniform Resource Locator, 23, 240

uruchamianie animacji, 344

usługa

- BrowserLab, 48
- Browsershots, 48
- BrowserStack, 48
- CrossBrowserTesting, 48
- dostarczania czcionek sieciowych, 142
- Google Fonts, 153, 157
- NetRenderer, 48

ustawianie

- cienia, 101
- marginesów, 224, 227
- obrazów w obramowaniu, 276
- odstępów, 224
- odstępów wokół znaczników, 227
- stałej szerokości, 417
- szerokości, 101
- tła, 224
- wartości pozycjonujących, 458
- wyrównania, 354

ustawienia strony, 182

usuwanie

- dopełnienia, 291
- elementów, 494
- elementów tła, 490
- marginesów, 174
- podkreślenia, 285
- punktorów, 291
- stylów przeglądarki, 509

używanie, *Patrz* stosowanie

W

W3C, World Wide Web Consortium, 42, 209, 563

waga stylu, 119

walidacja, 42

walidator W3C, 42, 55

warianty czcionek, 150

wartości, 51

- animation, 347

- background-origin, 246, 248

- background-repeat, 240

- box-sizing, 215, 409

- clear, 222

- CSS, 523

- float, 219, 389

- overflow, 216

- RGB, 524

- transition-timing-function, 328

wcięcie pierwszego wiersza, 175

wczytywanie plików z wyprzedzeniem, 299

wersje CSS, 27

wielkie litery, 168

wiersz tabeli, 358

witryny pokazowe, 568

własność, *Patrz* właściwość, 51

- właściwości
 - animacji, 551
 - CSS, 523–561
 - dodatkowe, 558
 - dziedziczone, 119, 120
 - list, 533
 - pozycjonujące, 456
 - przejść, 329, 551
 - przekształceń, 551
 - ramek, 204
 - tabel, 556
 - tekstu, 527
 - tła, 310
 - układu strony, 544
- właściwość, 51
 - animation, 342
 - animation-delay, 340
 - animation-direction, 341
 - animation-duration, 338
 - animation-fill-mode, 342
 - animation-iteration-count, 341
 - animation-name, 338
 - animation-play-state, 343
 - animation-timing-function, 339
 - background, 191, 249–251, 278, 490
 - background-attachment, 246
 - background-clip, 247
 - background-color, 264
 - background-image, 181, 236, 273, 289
 - background-origin, 246
 - background-position, 242, 277
 - background-repeat, 240, 253
 - background-size, 248
 - border, 205, 235, 275
 - border-collapse, 356, 367
 - border-radius, 207, 232, 357
 - border-spacing, 356
 - box-shadow, 161, 210, 232
 - box-sizing, 215, 409, 416
 - clear, 222, 397, 414, 449
 - display, 203, 491, 495
 - float, 190, 219, 271, 296, 393
 - font-family, 115, 135, 149
 - font-style, 149, 153
 - font-weight, 149, 153
 - height, 212
 - line-height, 173
 - list-style-image, 181
 - margin, 229, 236
 - max-width, 218, 419, 448
 - min-height, 218
 - opacity, 466, 479
 - overflow, 216
 - padding, 192, 228, 236
 - page-break-after, 493
 - page-break-before, 493
 - position, 347, 458
 - text-align, 354
 - text-decoration, 169
 - text-shadow, 161, 171
 - text-transform, 168, 187
 - transform, 317–325
 - transform-origin, 325
 - transition, 326, 332, 479
 - transition-delay, 331, 340
 - transition-duration, 327
 - transition-property, 327
 - transition-timing-function, 328, 339
 - vertical-align, 355
 - visibility, 466
 - width, 212, 437
 - z-index, 464
 - zbiorcza animation, 342
 - zbiorcza background, 249
 - zbiorcza transition, 332
 - zoom, 401, 410
- wojna na precyzję, 121, 123
- wolna przestrzeń, white space, 195
- wstawianie komentarzy, 501
- wstrzymywanie animacji, 343
- wtyczka aTip2, 468
- wybór czcionek, 137, 154
- wydruk strony, 500
- wymiary pól, 212, 214
- wyrównywanie tekstu, 174, 354
- wyróżnianie
 - akapitu, 189
 - odnośników, 306
- wyskakujące menu, 297
- wysokość
 - elementu, 213, 214
 - elementu pływającego, 402
 - kolumn, 407
 - minimalna, 218
- wyszukiwarka czcionek, 155
- wyświetlacze Retina, 164

wyświetlanie URL, 499
wyzwalacz, 326
wyzwalanie animacji, 337

X

XHTML, 24
XML, Extensible Markup Language, 24

Z

zalety CSS, 20
zamiana punktów, 277
zamienianie układu sztywnego, 437
zaokrąglanie rogów, 207, 357
zapytania o media, 427–434
zasoby CSS, 563
zastępowanie obramowania, 275
zatwierdzanie formularza, 372
zmienianie
 czarnego tła, 490
 formatowania tekstu, 498
 kolejności elementów, 443
 kolorów tła, 207, 345
 precyzji, 121
 rozmiaru pisma, 163
 stylów tekstu, 488
 stylu strony, 109
 wyglądu odnośnika, 283
zmienne CSS, 505
zmniejszanie strony, 426
znacznik, 21
 <a>, 23, 292
 <abbr>, 36
 <address>, 44, 65
 <article>, 38, 44, 382
 <aside>, 39, 411
 , 41, 153
 <blockquote>, 44
 <body>, 23

, 41
 <caption>, 366
 <cite>, 41
 <code>, 36
 <dd>, 44

 <div>, 37, 71, 380, 403
 <dl>, 44
 <fieldset>, 361
 <figcaption>, 468, 477
 <figure>, 39, 269, 477
 , 35
 <footer>, 24, 39, 382
 <h1>, 36
 <head>, 23
 <header>, 38, 40
 <hr>, 203
 <html>, 23, 42
 <i>, 153
 , 235
 <input>, 373
 <label>, 363
 <legend>, 361
 , 314
 <link>, 55
 <meta>, 426
 <nav>, 24, 284
 , 34, 44
 <p>, 23, 36
 <select>, 361
 <section>, 38, 44
 , 37, 44, 71, 381
 , 23, 96
 <style>, 59, 93
 <table>, 35, 354
 <td>, 87
 <textarea>, 361
 <th>, 366
 , 36, 43, 291
 otwierający, 22
 zamykający, 22
znaczniki, *Patrz także* elementy
 blokowe, 202
 HTML, 22, 36
 nagłówków, 34
 śródliniowe, 202
 zagnieżdżone, 74
znajdowanie czcionek, 154
zniekształcanie, 322
zwiększanie precyzji klasy, 122

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**

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>

sięgnij po WIĘCEJ



KOD KORZYŚCI

ISBN 978-83-246-7317-9



9 788324 673179

Cena: 79,00 zł

Informatyka w najlepszym wydaniu