

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

Po prostu DHTML i CSS

Autor: Jason Cranford Teague

Tłumaczenie: Agata Bulandra

ISBN: 83-7197-569-4

Tytuł oryginału: [DHTML and CSS for the World Wide Web, 2nd Edition, Visual QuickStart Guide](#)

Format: B5, stron: 592

[Przykłady na ftp: 7771 kB](#)



W książce tej pokazano najlepsze sposoby wprowadzania w życie DHTML-a i CSS, umożliwiające obejrzenie tych stron jak największej liczbie osób myśkujących po Internecie. W celu lepszej organizacji informacji książka jest podzielona na cztery części:

- Część 1. informuje, jak używać CSS, aby zapanować nad wyglądem zawartości stron internetowych. Pokażę właściwe sposoby kontrolowania różnych aspektów wyglądu strony.
- Część 2. zajmuje się sposobem wykorzystania obiektowego modelu dokumentu (DOM – Document Object Model) wraz z CSS i JavaScriptem do stworzenia prostych funkcji dynamicznych. Pokażę także, jak stworzyć DOM, pozwalający na uruchomienie funkcji dynamicznych w większości przeglądarek, maksymalnie ograniczając przy tym ilość zbędnego kodu.
- Część 3. skupia się nad sposobem wykorzystania DHTML-a i CSS w dwóch najbardziej znanych programach do edycji stron internetowych: Adobe GoLive i Macromedia Dreamweaver. Choć nie musimy używać tych programów przy tworzeniu stron internetowych w technologii DHTML i CSS, to mogą one ułatwić nasze życie.
- Część 4. pokazuje, jak przy użyciu DHTML-a i CSS zaprojektować witrynę internetową, opisuje także niektóre zastosowania tych technologii. Dodatkowo pokazuje, jak szukać błędów w kodzie oraz zapoznaje nas z dopiero pojawiającymi się technologiami.



Spis treści

	Wprowadzenie	13
Część I	Kaskadowe arkusze stylów	23
Rozdział 1.	Wprowadzenie do CSS	25
	Czym jest styl?	27
	Czym są kaskadowe arkusze stylów?	28
	Wersje CSS	31
	Typy reguł CSS	33
	Części składowe reguły CSS	34
	Rodzaje znaczników HTML	36
Rozdział 2.	Podstawy CSS	39
	Dodawanie CSS do znacznika HTML	40
	Dodawanie CSS do strony internetowej	42
	Dodawanie CSS do witryny internetowej	45
	(Re)Definiowanie znaczników HTML	51
	Definiowanie klas w celu stworzenia własnych znaczników	53
	Definiowanie identyfikatorów w celu identyfikacji obiektów	55
	Tworzenie własnych znaczników elementów wewnętrznych	57
	Tworzenie własnych znaczników elementów blokowych	59
	Definiowanie znaczników z tą samą regułą	61
	Definiowanie znaczników w kontekście	63
	Tworzenie definicji !important	65
	Dziedziczenie właściwości od rodzica	67
	Zarządzanie istniejącymi i dziedziczonymi wartościami właściwości	69
	Ustalanie kolejności w kaskadzie	71
	Tworzenie arkusza stylów dla wydruków	73
	Dodawanie komentarzy	76
	Strategie arkuszy stylów	77

Rozdział 3.	Sterowanie czcionkami	79
	Podstawy typografii używanej w Sieci	80
	Rodzaje prezentacji tekstu w Sieci	81
	Ustawianie czcionki	83
	Pobieranie czcionek	85
	Używanie czcionek właściwych dla przeglądarki	87
	Ustawianie wielkości czcionki	89
	Ustawianie kursywy	91
	Gruby, grubszy, najgrubszy	93
	Tworzenie kapitalików	95
	Ustawianie kilku wartości dla czcionki	96
Rozdział 4.	Kontrolowanie tekstu	99
	Dostosowywanie kerningu	100
	Dostosowywanie odstępów między słowami	101
	Dostosowywanie interlinii	102
	Ustawianie wielkości liter	104
	Wyśrodkowanie tekstu oraz wyrównanie do lewej, prawej strony	105
	Pionowe wyrównanie tekstu	107
	Tworzenie wcięć akapitów	109
	Dekorowanie tekstu	111
	Kontrolowanie pustej przestrzeni	113
	Ustawianie podziału strony przy drukowaniu	115
Rozdział 5.	Kontrolowanie list i kursorów	117
	Tworzenie listy	118
	Ustawianie stylów znaków wypunktowania	120
	Tworzenie wcięcia list	121
	Tworzenie własnych znaków wypunktowania	122
	Zmiana wyglądu kursora myszy	124
Rozdział 6.	Kontrolowanie koloru i tła	127
	Ustawianie tła	128
	Ustawianie poszczególnych właściwości tła	131
	Określanie koloru pierwszego planu	136

Rozdział 7.	Kontrolowanie ramek i marginesów	137
	Pudełko elementu — podstawowe informacje	138
	Ustawianie szerokości i wysokości elementu	140
	Ustawianie marginesów elementu	143
	Ustawianie ramek elementu	146
	Ozdabianie ramek elementu	148
	Ustawianie dopełnienia elementu	151
	Ustawianie ramek i marginesów tabeli	153
	Oblewanie elementu tekstem	155
	Uniemożliwianie oblewania tekstem	157
	Określanie sposobu wyświetlenia elementu	159
Rozdział 8.	Kontrola pozycjonowania	161
	Okno — podstawowe informacje	162
	Ustawianie typu pozycjonowania	164
	Ustawianie pozycji lewej i górnej	168
	Ustawianie pozycji prawej i dolnej	171
	Układanie obiektów w stos (pozycjonowanie trójwymiarowe)	173
	Zagnieżdżanie elementu pozycjonowanego bezwzględnie w elemencie pozycjonowanym względnie	175
	Zagnieżdżanie elementu pozycjonowanego względnie w elemencie pozycjonowanym bezwzględnie	177
Rozdział 9.	Kontrolowanie widzialności	179
	Ustawianie widzialności elementu	180
	Ustawianie widocznego obszaru elementu (przycinanie)	182
	Nadzór nad sposobem wyświetlenia pozostałej zawartości	184
Część II	Dynamiczny HTML	187
Rozdział 10.	DHTML — podstawowe informacje	189
	Czym jest dynamiczny HTML?	190
	Smaczki DHTML-a	191
	Dlaczego powinniśmy używać DHTML-a?	193
	Flash kontra DHTML	195

Rozdział 11. Obiektowy model dokumentu	199
DOM — plan strony internetowej.....	200
Tworzenie obiektu.....	202
Procedury obsługi zdarzeń	203
Wykrywanie zdarzenia	205
Jak działa DOM.....	207
Sprawdzanie cech.....	213
Sprawdzanie typu DOM.....	214
Tworzenie DOM zgodnego z różnymi przeglądarkami	216
Używanie DOM zgodnego z różnymi przeglądarkami	219
Netscape Navigator 4 a zagnieżdżane warstwy.....	221
Rozdział 12. Zbieranie informacji na temat środowiska	225
Sprawdzanie nazwy i wersji przeglądarki	226
Sprawdzanie systemu operacyjnego.....	228
Sprawdzanie wymiarów ekranu	230
Sprawdzanie ustawionej liczby kolorów	232
Sprawdzanie wymiarów okna przeglądarki	234
Sprawdzanie wymiarów widocznej części strony	235
Sprawdzanie tytułu i adresu strony	237
Sprawdzanie pozycji, do której została przewinięta strona.....	238
Sprawdzanie wymiarów obiektu	240
Sprawdzanie górnej i lewej pozycji obiektu.....	242
Sprawdzanie dolnej i prawej pozycji obiektu.....	244
Sprawdzanie kolejności nakładania się obiektów	246
Sprawdzanie stanu widzialności obiektu.....	248
Sprawdzanie widocznego obszaru obiektu.....	250
Rozdział 13. Podstawowe techniki dynamiczne	255
Wyświetlanie i ukrywanie obiektów	256
Przesuwanie obiektu z miejsca na miejsce.....	258
Przesunięcie obiektu o pewną odległość	260
Zmiana wartości indeksu Z elementów	262
Przewijanie strony internetowej	265
Zmiana widocznego obszaru obiektu	267

Rozdział 14. Zaawansowane techniki dynamiczne	269
Powtórne uruchomienie funkcji	270
Przekazanie wydarzenia do funkcji	273
Globalna procedura obsługi zdarzeń	274
Animowanie obiektu	276
Sprawdzanie umiejscowienia wskaźnika na ekranie	279
Rozpoznanie elementu na ekranie	281
Współpraca dynamicznej zawartości różnych ramek	283
Przesuwanie okna przeglądarki	286
Otwieranie nowego okna przeglądarki	288
Zmiana rozmiaru okna	292
Rozdział 15. Techniki dynamiczne — CSS	295
Zmiana deklaracji	296
Zmiana klasy obiektu	298
Dodawanie nowej reguły	300
Unieważnianie arkusza stylów	302
Rozdział 16. Warstwy Netscape	305
Czym są warstwy Netscape?	306
Tworzenie warstwy	307
Importowanie zewnętrznej zawartości za pomocą warstw Netscape	310
Uzyskiwanie dostępu do warstw przy użyciu JavaScriptu	312
Modyfikowanie warstw za pomocą JavaScriptu	316
Przeglądarki nie obsługujące warstw — umieszczanie treści	318
Rozdział 17. Internet Explorer dla Windows	319
Przenikanie obiektu w obiekt	320
Przejsięcie pomiędzy stronami	321
Rozmywanie elementu	323
Obiekty „falujące”	324
Część III Używanie narzędzi DHTML i CSS	325
Rozdział 18. Elementarz GoLive	327
Interfejs GoLive	328
Dodawanie CSS	332
Dodawanie warstwy (pływającego pudełka)	336
Dodawanie animacji DHTML	338

Rozdział 19.	Elementarz Dreamweavera	341
	Interfejs Dreamweavera	342
	Dodawanie CSS.....	346
	Dodawanie warstwy	350
	Dodawanie animacji.....	352
Część IV	Dynamiczne witryny internetowe	355
Rozdział 20.	Podstawy dynamicznej Sieci	357
	Co sprawia, że witryna jest dynamiczna?	358
	Co to jest hipertekst?	360
	Dynamicznie poprzez projekt.....	361
	Układ strony — podstawowe informacje	362
	Tworzenie nawigacji — o czym powinniśmy pamiętać, a czego powinniśmy unikać	365
Rozdział 21.	Tworzenie dynamicznej witryny internetowej	369
	Pierwszy krok — określ	370
	Drugi krok — zaprojektuj	373
	Trzeci krok — zbuduj	378
Rozdział 22.	Układ strony internetowej	381
	Naprawianie błędu implementacji CSS w Netscape Navigatorze.....	382
	Tworzenie arkusza stylów dla różnych systemów operacyjnych.....	384
	Tworzenie nagłówków	386
	Tworzenie nagłówka o ustalonej pozycji	388
	Tworzenie bocznego menu.....	390
	Ustawianie dynamicznego nagłówka i stopki	392
	Tworzenie własnych obramowań ramek.....	394
	Otwieranie i zamykanie ramek.....	396
	Zachowanie podziału strony na ramki.....	402
	Dobry wygląd w druku (w Sieci)	405
Rozdział 23.	Importowanie zewnętrznej zawartości	407
	Łączenie znaczników <ilayer> i <iframe>	408
	Dołączanie po stronie serwera.....	410
	Używanie zewnętrznych plików JavaScript.....	411
	Przeglądanie zewnętrznych skryptów innych autorów	412

Rozdział 24. Nawigacja po witrynie	415
Ustawianie stylów odnośników	416
Ustawianie wielu stylów odnośników	419
Tworzenie rozwijanych list	421
Tworzenie wysuwających się menu	426
Tworzenie zdalnego sterowania	429
Tworzenie rozwijanych menu	433
Tworzenie wielopoziomowego menu	437
Nawigacja w przeglądarkach niedynamicznych	443
Informowanie przeglądarki	444
Rozdział 25. Obiekty sterujące	447
Tworzenie własnych suwaków	448
Tworzenie własnego przycisku Wstecz	455
Tworzenie pokazu slajdów	456
Tworzenie wyskakującego hipertekstu	460
Używanie danych z formularzy w dynamicznych akcjach	463
Tworzenie formularzy kontekstowych	465
Przeciąganie i upuszczanie obiektów	468
Wymiana obrazków	471
Tworzenie inteligentnych menu	476
Rozdział 26. Efekty specjalne	481
Tworzenie inicjałów	482
Tworzenie nieskomplikowanych cieni	484
Tworzenie zaawansowanych cieni	486
Pojawianie się lub wygaszanie tekstu HTML	489
Podążaj za kursorem myszy	493
Pływające obiekty	497
Tworzenie przezroczystej grafiki w formacie PNG	500
Tworzenie zegara	505
Rozdział 27. Multimedia	507
Dodawanie dźwięku	508
Dodawanie animacji GIF	510
Dodawanie animacji Flash	517
Dodawanie filmów	523
Dodawanie apletów Javy	526

Rozdział 28.	Usuwanie błędów w kodzie	529
	Identyfikowanie i usuwanie usterek CSS	530
	Sprawdzanie kodu CSS	533
	Identyfikowanie i usuwanie usterek JavaScript	535
	Międzyprzeglądarkowe zagadki	539
Rozdział 29.	Przyszłość dynamicznej Sieci	541
	Dlaczego standardy są ważne?	542
	Rozszerzalny język znakowania (XML)	544
	Rozszerzalny hipertekstowy język znakowania (XHTML)	547
	Język zsynchronizowanej integracji multimediiów (SMIL)	551
	Skalowalna grafika wektorowa (SVG).....	553
	Co dalej: CSS poziom 3.	554
Dodatki		555
Dodatek A	Przeglądarki obsługujące DHTML-a i CSS	557
	Internet Explorer	558
	Netscape Navigator	559
Dodatek B	Przeglądarki obsługujące DHTML-a i CSS	561
	Krótki przegląd.....	562
Dodatek C	DHTML — krótka ściągga	571
	Słowa zastrzeżone	574
	Skorowidz	577

Układ strony internetowej

22

Jest tak wiele sposobów wykorzystania CSS i DHTML-a, jak wielu jest projektantów używających tych technologii. Narzędzia te są stosunkowo nowe i projektanci wciąż odkrywają ich możliwości i ograniczenia. Dodatkowo niektórzy projektanci, początkowo zafascynowani możliwościami CSS w tworzeniu dynamicznego HTML-a, zlekceważyli wiele z jego zalet, dotyczących układu strony.

Spiesząc się do eksperymentów z dynamicznymi aspektami CSS, wielu projektantów przeoczyło pewne praktyczne szczegóły problemów, które CSS rozwiązuje. CSS ułatwia stworzenie jednolitego układu strony — po prostu nie do odrzucenia.

Rozdział ten odkrywa niektóre z cennych rozwiązań CSS, oferowanych do rozwiązania codziennych problemów związanych z projektem i — jako najlepszy sposób zintegrowania DHTML-a — z układem stron.

Naprawianie błędów implementacji CSS w Netscape Navigatorze

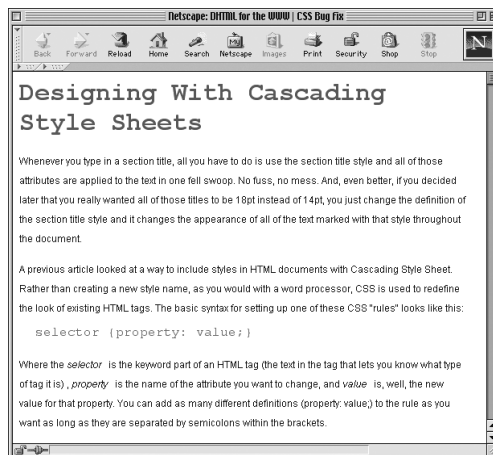
W Netscape Navigatorze istnieje jeden oczywisty błąd związany z CSS, na który autorzy bardzo często narzekają. Gdy odwiedzający zmieni wielkość okna przeglądarki, wszystkie style CSS, pochodzące z zewnętrznego pliku arkusza stylów (podłączanego poprzez znacznik `<link>`), tajemniczo znikają, tak jakby podłączony plik CSS w ogóle nie istniał. Jednak jeśli odwiedzający ponownie załaduje stronę, to style CSS pojawiają się znowu (rysunki 22.1 i 22.2). Błąd ten może być czymś odpychającym dla odwiedzającego stronę, zwłaszcza jeśli nie wie, że ponowne załadowanie strony rozwiązuje ten problem.

Jak upewnić się, że strona zostanie przeładowana po zmianie wielkości okna? Wystarczy sprawić, by przeglądarka zwracała uwagę na to, co dzieje się w jej środowisku (kod 22.1).

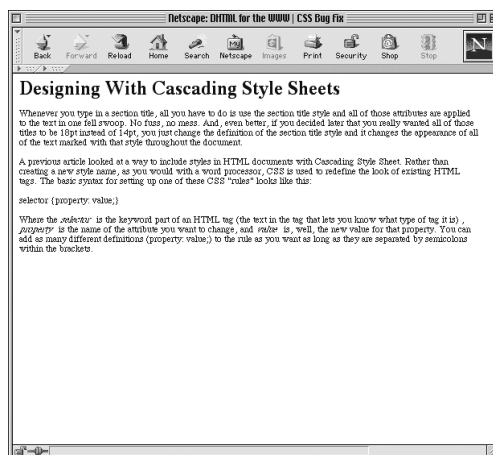
Aby wymusić ponowne załadowanie strony po zmianie jej wielkości:

1. `if (document.layers) {...}`

W znaczniku `<script>` w nagłówku strony HTML (`<head>`) dodaj kod wykrywający, czy przeglądarka korzysta z obiektowego modelu dokumentu dla warstw (zobacz *Sprawdzanie cech* w rozdziale 11.). Jeśli go używa, kod zarejestruje aktualną szerokość (`innerWidth`) i wysokość (`innerHeight`) widocznego obszaru strony (zobacz *Sprawdzanie wymiarów widocznej części strony* w rozdziale 12.).



Rysunek 22.1. Tak strona powinna wyglądać w Netscape Navigatorze



Rysunek 22.2. Bez kodu naprawiającego błąd w Netscape Navigatorze, gdy odwiedzający zmieni wielkość ekranu, do wyświetlenia strony zostają użyte ustawienia domyślne

Kod 22.1. Jeśli przeglądarka korzysta z warstw (co może oznaczać, że jest Netscape Navigatorem 4), JavaScript rejestruje początkową wartość szerokości i wysokości dostępnego obszaru okna przeglądarki. Jeśli wielkość okna ulegnie zmianie, kod porówna początkowe wymiary z nowymi i — jeśli się różnią — strona zostaje przeładowana, co przywraca zastosowanie zewnętrznego pliku CSS

```
Kod
<html>
<head>
  <script>
    if (document.layers) {
      origWidth = innerWidth;
      origHeight = innerHeight;
    }
    function reloadPage() {
      if (innerWidth != origWidth ||
      innerHeight != origHeight)
        location.reload();
    }
    if (document.layers) onresize = reloadPage;
  </script>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>Designing With Cascading Style
  Sheets</h1>
  <p class="copy">Whenever you type in a
  section title...</p>
  <p class="copy">A previous article looked
  at a...</p>
  <div class="code">
    selector {property: value;}<BR>
  </div>
  <p class="copy">Where the <I>selector</I>
  is the keyword part of an HTML...</p>
</body>
</html>
```

2. function reloadPage() {...}

Do skryptu dodaj funkcję reloadPage(). Funkcja, gdy zostanie wywołana, porówna obecną wysokość i szerokość widocznej części strony z wartościami zarejestrowanymi w punkcie 1. Jeśli wartości się różnią, strona zostanie przeładowana.

3. onresize = reloadPage;

Dodaj procedurę onresize, aby uruchomić funkcję reloadPage z punktu 2. Jeśli użytkownik zmieni wielkość okna, zmieniając tym samym widoczny obszar, to przeglądarka ponownie załaduje stronę, przywracając działanie zewnętrznego arkusza stylów CSS.

Wskazówka

- Dobrym pomysłem jest umieszczenie tego kodu w zewnętrznym pliku .js, który będziemy importować do każdej strony w witrynie, wykorzystującej zewnętrzny plik CSS (zobacz *Używanie zewnętrznych plików JavaScript* w rozdziale 23.).

Tworzenie arkuszy stylów dla różnych systemów operacyjnych

Projektantów tworzących za pomocą CSS denerwuje kilka niezgodności pomiędzy systemami Windows i MacOS, zwłaszcza w przypadku ustalania wielkości czcionki i kolorów. Tak naprawdę sam problem tkwi nie w CSS, ale w sposobie, w jaki te systemy operacyjne definiują wielkość czcionki i kolor na ekranie.

Bez zagłębiania się w szczegóły historyczne i techniczne — podstawowy problem polega na tym, że Windows ten sam rozmiar czcionki wyświetla jako większy, niż robi to MacOS, a ten sam kolor wyświetla jako nieco ciemniejszy. Taka sytuacja może sprawić, że projekt będzie wyglądał wyśmianie na MacOS, ale będzie miał zbyt duży tekst i zbyt ciemne kolory na komputerze typu PC.

Jaka jest na to rada? Wykorzystując JavaScript i kilka arkuszy stylów dopasowanych do kolejnych systemów operacyjnych, można każdemu z nich udostępnić odpowiednie rozmiary czcionki i kolor (rysunki 22.3, 22.4 i 22.5).

Aby stworzyć CSS odpowiedni dla systemu operacyjnego odwiedzającego:

1. default.css

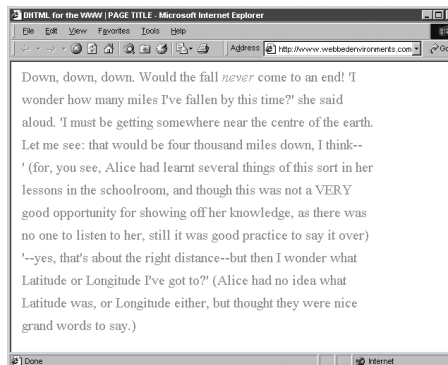
Stwórz zewnętrzny plik CSS, zawierający style, które zostaną użyte w witrynie internetowej i zapamiętaj go jako *default.css* (kod 22.2). Plik ten zostanie bezpośrednio podłączony do stron w punkcie 3.

2. mac.css

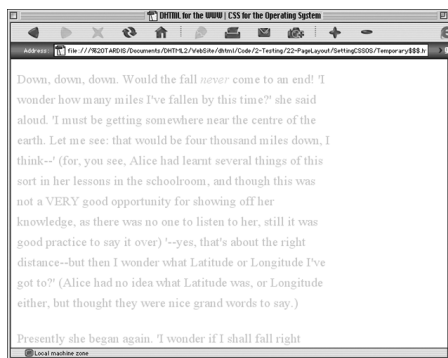
Stwórz drugi plik z arkuszem stylów i zapamiętaj go jako *mac.css* (kod 22.3). Ta wersja pliku powinna zostać wykorzystana do podmiany deklaracji z pliku *default.css* i dostosowania ich dla użytkowników systemu MacOS poprzez zwiększenie czcionki i dopasowanie koloru. Nie trzeba powtarzać wszystkich deklaracji z pliku *default.css*, ponieważ te, których chcemy użyć, znajdują się w kaskadzie arkuszy stylów.

3. <link href="default.css" rel="stylesheet" type="text/css">

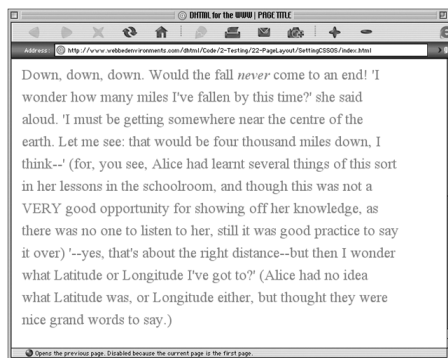
W nagłówku dokumentu HTML — w tym przykładzie *index.html* (kod 22.4) — podłącz domyślną wersję arkusza stylów.



Rysunek 22.3. Strona wyświetlona w Internet Explorerze 5 dla Windows



Rysunek 22.4. Ta sama strona wyświetlona w Internet Explorerze 5 dla systemu MacOS (bez korekcji). Tekst jest mniejszy i zbyt jasny



Rysunek 22.5. Ta sama strona wyświetlona w Internet Explorerze 5 dla systemu MacOS z korekcyjnym arkuszem stylów. Aby zrekompensować wcześniejsze błędy, tekst jest nieco większy i ciemniejszy

Kod 22.2. Domyślny arkusz stylów *default.css* zawiera wszystkie domyślne style, które powinny zostać użyte na stronie

```
Kod
.copy {
  color: #cc3;
  font: 20px/32px Times New Roman,
  Georgia, Times, serif;
  width: 500px;
}

body {
  background-color: #fff;
}
```

Kod 22.3. Style w pliku *mac.css* unieważniają odpowiednie style z pliku *default.css*

```
Kod
.copy {
  color: #bb2;
  font: 23px/35px;
  width: 600px;
}
```

Kod 22.4. JavaScript. System, w którym działa przeglądarka, to MacOS. Jeśli tak, to kolejny arkusz stylów zostaje dodany do strony i wprowadza pewne korekcie

```
Kod
<html>
<head>
  <link href="default.css" rel="stylesheet"
  type="text/css">
  <script language="JavaScript">
  if ((navigator.appVersion.indexOf('Mac') != -
  1)) {
  document.write('<link href="mac.css"
  rel="stylesheet" type="text/css">'); }
  </script>
</head>
<body>
  <p class="copy">Down, down, down...</p>
  <p class="copy">Presently she began
  again...</p>
</body>
</html>
```

4. if ((navigator.appVersion.indexOf('Mac') != -1)) {...}

Za znacznikiem `<link>` umieść skrypt JavaScript, który sprawdza, czy przeglądarka wyświetlająca stronę jest używana w środowisku MacOS. Jeśli jest, to znacznik `<link>` podłączający wersję arkusza stylów dla MacOS zostaje „dopisany” do strony poprzez JavaScript. Drugi arkusz stylów dopasowuje deklaracje tak, by były odpowiednie dla systemu MacOS.

Wskazówki

- Chociaż wersja stylu CSS dla klasy `copy` w MacOS nie zawiera deklaracji czcionki, to tekst i tak wyświetlony jest czcionką Times. Dlaczego reguła dla klasy `copy` z pliku CSS dla MacOS nie podменя deklaracji w domyślnym pliku CSS? Pojęcie *kaskadowe* w *kaskadowych* arkuszach stylów odnosi się do możliwości mieszania deklaracji, nawet gdy pochodzą z różnych źródeł.
- Systemy operacyjne to nie jedyne źródło problemów. Przeglądarki, nawet gdy działają w tym samym systemie operacyjnym, także dziwnie wyświetlają rozmiar czcionki i pozycjonują elementy na stronie. Aby wykręcić typ przeglądarki, można użyć tej samej techniki (zobacz *Sprawdzanie nazwy i wersji przeglądarki* w rozdziale 12.) i udostępnić każdej przeglądarce odpowiedni arkusz stylów.
- Rozwiązania JavaScript pokazanego w tym podrozdziale można użyć także w innych celach. Jeśli chcemy udostępnić różne arkusze stylów w zależności od preferencji odwiedzającego, można użyć *cookie*, aby nadzorować, jakie arkusze stylów są ładowane. Skrypt umożliwia projektantowi i odwiedzającemu stronę dużo większą kontrolę nad tym, jak wyświetlona jest strona, a projektant nie musi tworzyć nowych stron dla każdej wersji.

Tworzenie nagłówków

Kolejna rzecz, irytująca projektantów, to tworzenie nagłówków za pomocą grafiki, co najczęściej oznacza tworzenie nowej grafiki dla każdego nagłówka. Wykorzystując właściwość tła CSS, można stworzyć wiele różnych tytułów bez konieczności tworzenia nowego pliku dla każdego z nich. Takie rozwiązanie ma dodatkową zaletę, polegającą na tym, że nie wydłużamy czasu pobierania strony przez umieszczenie tekstu w plikach graficznych.

Aby stworzyć nagłówek z graficznym tłem:

1. background_headline.gif

W programie graficznym stwórz i zapamiętaj tło. Nazwij ten plik np. background_headline.gif (rysunek 22.6).

2. h3.graphic {...}

Dodaj regułę CSS dla znacznika <h3> (kod 22.5) z przypisaną klasą graphic (zobacz *Definiowanie klas w celu stworzenia własnych znaczników* w rozdziale 2.). Dołącz właściwość tła i zadeklaruj użycie grafiki stworzonej w punkcie 1. (zobacz *Ustawianie tła* w rozdziale 6.).

Uwaga: Klasy zdefiniowanej w punkcie 2. nie trzeba nazywać graphic; możemy użyć dowolnej innej nazwy.

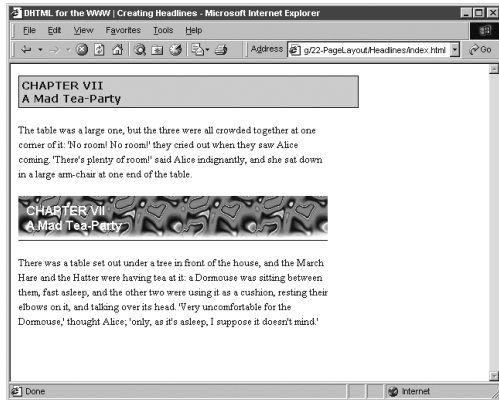


Rysunek 22.6. Grafika, która wypełni tło nagłówków

Kod 22.5. Zastosowanie grafiki w tle nagłówków jest proste, a możliwości są nieskończone

```
Kod
<html>
<head>
  <style type="text/css">
    h3.offset {
      color: #000000;
      font-weight: bold;
      font-size: 14px;
      font-family: Verdana, Arial, Helvetica,
        sans-serif;
      background-color: #ccc;
      padding: 3px;
      border: solid 1pt #000000;
      position: relative;
      width: 440px;}
    h3.graphic {
      background: black
      url(background_headline.gif) no-repeat;
      font: bold 16px helvetica,sans-serif;
      color: white;
      width: 400px;
      padding:10px; }
    p {
      font: normal 10pt/14pt times,serif;
      left-margin: 25px;
      width: 400; }
  </style>
</head>
<body>
  <h3 class="offset">CHAPTER VII<br>
  A Mad Tea-Party</h3>
  <p>The table was a large one...</p>

  <h3 class="graphic">CHAPTER VII<br>
  A Mad Tea-Party</h3>
  <p>There was a table set out under a tree
  in front of the house...</p>
</body>
</html>
```



Rysunek 22.7. Dwa przykłady nagłówka. W celu uzyskania innych efektów można poeksperymentować z różnymi grafikami, ramkami, a nawet z innym dopełnieniem



Rysunek 22.8. Obydwa nagłówki („About the Site” i „Reading the Code”) są tekstem HTML z grafiką w tle

3. `<h3 class="graphic">CHAPTER VII<br>A Mad Tea-Party</h3>`

Od tej chwili zawsze, gdy będziemy używać w dokumencie nagłówka trzeciego stopnia i dodamy do niego atrybut `class` z klasą, którą zdefiniowaliśmy w punkcie 2., to w jego tle zostanie wyświetlona grafika (rysunek 22.7).

Wskazówki

- Tło dla pozostałych poziomów nagłówków można stworzyć w taki sam sposób. Można użyć dla nich tej samej lub innej grafiki poprzez zgrupowanie selektorów (zobacz Definiowanie znaczników z tą samą regułą w rozdziale 2.).
- Można eksperymentować z różnymi grafikami w tle. Grafiką dla tła, które stworzyłem dla jednej ze swych witryn, był gradient, który zaczynał się pewnym kolorem po lewej i stopniowo przechodził w kolor tła po prawej stronie (rysunek 22.8).

Tworzenie nagłówka o ustalonej pozycji

Najważniejszą zasadą dobrego projektu jest informowanie ludzi o tym, gdzie się znajdują. Niestety, strony się przewijają, a ważne informacje na temat oglądanej strony (takie jak na przykład tytuł strony) zazwyczaj przewijają się na samą górę i stają się niewidoczne.

Wykorzystując CSS, można ustalić położenie tytułu na górze strony tak, by był zawsze widoczny i informował, w którym miejscu witryny znajdują się odwiedzający — bez względu na to, jak została przewinięta strona (rysunek 22.9).

Jednak trzeba mieć na uwadze, że Internet Explorer 4 (dla Windows i MacOS) i 5 (dla Windows) oraz Netscape Navigator 4 i 6 (dla Windows i MacOS) nie obsługują ustalonej pozycji. Jedyna przeglądarka, która obsługuje ten typ pozycjonowania, to Internet Explorer 5 dla systemu MacOS.

Aby stworzyć nagłówek o ustalonej pozycji:

1. #header {

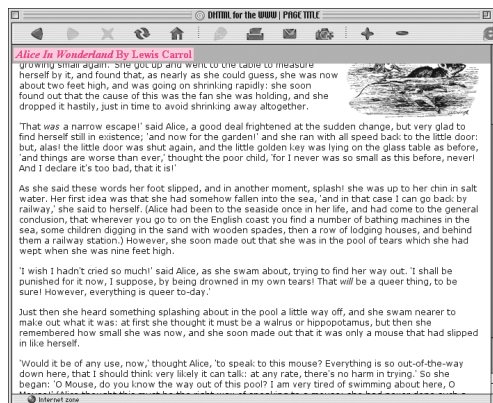
Rozpocznij listę deklaracji selektorem klasy lub identyfikatora. W tym przykładzie stworzyłem identyfikator o nazwie header (kod 22.6).

2. position: fixed;

Wpisz nazwę właściwości position i przypisz jej wartość fixed.

3. color: red;

Dodaj do listy pozostałe deklaracje, które chcesz zastosować przy tworzeniu nagłówka. W tym przykładzie zostanie wyświetlony czerwony tekst na szarym tle.



Rysunek 22.9. Pomimo że tekst został przewinięty, nagłówek pozostał u góry okna przeglądarki

Kod 22.6. Styl nagłówka o ustalonej pozycji został przypisany do identyfikatora, który został następnie nadany znacznikowi `<div>`

```
Kod
<html>
<head>
  <style type="text/css">
#header {
  color: red;
  font-weight: bold;
  font-size: 16px;
  font-family: "Times New Roman", Georgia,
Times, serif;
  background-color: #aaa;
  padding: 5px;
  position: fixed;
  z-index: 1000;
  top: 0px;
  left: 0px;
  width: 110%;
  visibility: visible; }
</style>
</head>
<body>
  <div id="header">
    <i>Alice In Wonderland</i> By Lewis
Carroll
  </div>
  <br>
  <p>'I'm sure those are not the right
words...</p>
  <p>As she
said this she looked down at her hands...</p>
  <p>'That <i>was</i> a narrow
escape!...</p>
  <p>As she said these words her foot
slipped...</p>
</body>
</html>
```

4. }

Zakończ listę deklaracji zamykającym nawiasem klamrowym (}).

5. `<div id="header"><i>Alice In wonderland</i> By Lewis Carroll</div>`

Dodaj identyfikator do odpowiedniego elementu. W tym przypadku do stworzenia tytułu strony wykorzystałem znacznik `<div>`.

Wskazówki

- Należy pamiętać, że ta technika nie będzie działała we wszystkich przeglądarkach. Przeglądarki nie rozpoznające tego typu pozycjonowania traktują nagłówki tak, jakby był elementem statycznym i przewijają go wraz z pozostałą zawartością strony. Pozostała część formatowania CSS będzie działać.
- Chociaż byłoby wspaniale, gdyby można było umieszczać odnośniki w nagłówku o niezmienniej pozycji, to błąd w Internet Explorerze 5 dla MacOS powoduje, że elementy o ustalonej pozycji są prawie bezużyteczne (zobacz notatkę *Czy jest niezmienna?* w rozdziale 8.).

Tworzenie bocznego menu

Boczne menu są chyba najczęściej stosowanym sposobem umieszczania narzędzi do nawigacji. Tradycyjnie tworzone szeroką grafikę tła, zawierającą tło bocznego menu i tło obszaru zawartości. Jednak wykorzystując CSS, wystarczy stworzyć dużo mniejszą grafikę, mającą szerokość wyłączenie bocznego menu i użyć wartości `repeat-y`, aby wypełnić tło bocznego menu obrazkiem powielanym w pionie przy lewym brzegu okna przeglądarki. Technika ta zmniejsza czas pobierania strony poprzez zmniejszenie wielkości pliku grafiki tła.

Aby stworzyć boczne menu:

1. `background_side.gif`

Stwórz grafikę — cienki pasek o szerokości takiej, jaka jest przeznaczona dla menu (rysunek 22.10).

2. `body { background: #cccccc url (background_side.gif) repeat-y; }`

W dokumencie stwórz regułę dla selektora znacznika `<body>` i zadeklaruj wykorzystanie grafiki utworzonej w punkcie 1. oraz to, że powinna być powielana jedynie wzdłuż osi Y (zobacz *Ustawianie poszczególnych właściwości tła* w rozdziale 6.). Także tutaj można zadeklarować kolor tła dla pozostałej części strony (kod 22.7).

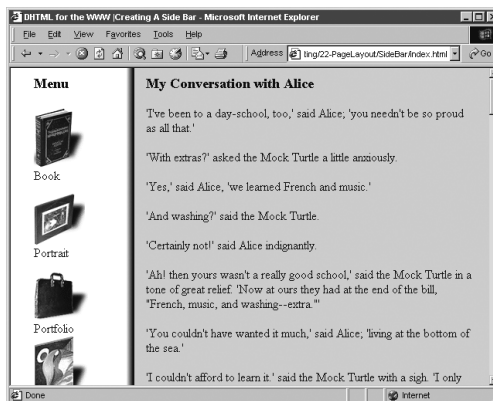
3. Wykorzystaj tabele lub pozycjonowanie CSS, aby stworzyć lewą kolumnę, w której znajdzie się zawartość bocznego menu, i prawą kolumnę dla pozostałej treści.

W tym przykładzie stworzyłem dwie pozycjonowane bezwzględnie kolumny, jedną z przeznaczeniem na menu, a drugą na zawartość (rysunek 22.11).

Wskazówka

- Można eksperymentować z różnymi grafikami i różnymi efektami. Można wypróbować powtarzanie grafiki w poziomie (`repeat-x`), aby zamiast bocznego menu stworzyć nagłówki. Możliwości są nieskończone.

Rysunek 22.10. Grafika tła wykorzystana do stworzenia bocznego menu o szerokości 160 pikseli



Rysunek 22.11. Grafika bocznego menu wypełnia lewą część ekranu do samego dołu strony, a kolor tła wypełni pozostałą część ekranu po prawej stronie

Kod 22.7. Stwórz boczne menu, używając grafiki tła i tworząc na stronie dwie kolumny

```

Kod
<html>
<head>
  <style type="text/css">
body {
  background: #cccccc url(background_side.gif) repeat-y;}
#sidebar {
  position: absolute;
  top: 10px;
  left: 30px;
  width: 150px; }
#content {
  position: absolute;
  top: 10px;
  left: 175px;}
  </style>
</head>
<body>
  <div id="sidebar">
    <h3>Menu</h3>
    <br>Book<br>
    <br>Portrait<br>
    <br>Portfolio<br>
    <br>Painting<br>
    <br>Letter<br>
     <br>Tools
  </div>
  <div id="content">
    <H3>My Conversation with Alice</H3>
    <p>'I've been to a day-school, too,' said Alice; 'you needn't be so proud as all
that.'</p>
    <p>'With extras?' asked the Mock Turtle a little anxiously.</p>
    <p>'Yes,' said Alice, 'we learned French and music.'</p>
    <p>'And washing?' said the Mock Turtle.</p>
    <p>'Certainly not!' said Alice indignantly.</p>
  </div>
</body>
</html>

```

Ustawianie dynamicznego nagłówka i stopki

Z dużymi witrynami wiąże się jeden problem — trudno jest zmienić ich wygląd, gdy już się je utworzyło.

Moja witryna prezentuje około stu artykułów, które napisałem na przestrzeni kilku lat. Nie jestem guru baz danych, dlatego przechowuję je jako statyczne strony HTML, ale zawsze lubiłem od czasu do czasu zmienić wygląd witryny. A więc, zamiast umieszczać nagłówki i stopki bezpośrednio na stronie, wykorzystuję JavaScript do zapisania ich w odpowiednim miejscu. Aby przekazać tytuł, datę i inne informacje na temat artykułu do zewnętrznego skryptu, wykorzystuję zmienne (zobacz *Używanie zewnętrznych plików JavaScript* w rozdziale 23.).

Aby stworzyć dynamiczny nagłówek i stopkę:

1. header.js

Stwórz zewnętrzny plik JavaScript i zapisz go jako header.js. Plik w punkcie 4. zostanie zaimportowany na górę strony index.html. W pliku dodaj metody document.writeln(...), aby zapisać cały kod HTML nagłówka strony. Dodatkowo nagłówek będzie używał zmiennych (index.html), aby dodać tytuł, podtytuł, sentencję i datę (kod 22.8).

2. footer.js

Stwórz zewnętrzny plik JavaScript i zapisz go jako footer.js. Plik w punkcie 4. zostanie zaimportowany na dół strony index.html. W pliku dodaj metody document.writeln(...), aby zapisać cały kod HTML stopki strony. W tym przykładzie stopka wyświetla tytuł strony (ten znajdujący się pomiędzy znacznikami <title> w nagłówku dokumentu, a nie wartość zmiennej title) oraz adres URL strony, a także odnośnik do strony o prawach autorskich oraz odnośnik mailto (kod 22.9).

Kod 22.8. Ten plik JavaScript jest importowany na górę dokumentu index.html, aby stworzyć nagłówek dokumentu

```
Kod
document.writeln ('<h1>')
document.writeln (title)
document.writeln ('</h1>')
document.writeln ('<h3>')
document.writeln (subTitle)
document.writeln ('</h3>')
document.writeln ('<i>')
document.writeln (teaser)
document.writeln ('</i><br><br> <span
style="font: 10pt arial">')
document.writeln (dDate + ' ' + mDate + ' ' +
yDate)
document.writeln ('</span>')
```

Kod 22.9. Ten plik JavaScript jest importowany na dół dokumentu index.html, aby stworzyć stopkę dokumentu

```
Kod
document.writeln('<br><hr><br clear="all">')
document.writeln('<span class="copyright">')
document.writeln('<b>Title:</b> ' +
self.document.title)
document.writeln('</br>')
document.writeln('<b>URL:</b> <a href="' +
self.location + '">' + self.location + '</a>'
)
document.writeln('</br>')
document.writeln('<a
href="../aux/copyright.html">Copyright
&copy;</a> 1998-2000 <A
HREF="mailto:jason@webbedenvironments.com">Ja
son Cranford Teague</A>. All rights
reserved.' )
document.writeln('</span></br>')
```

Kod 22.10. Oto przykładowa strona internetowa (*index.html*), importująca nagłówki i stopkę. Zawiera także kilka zmiennych JavaScript, dodających do nagłówka dokumentu tytuł, podtytuł, sentencję i datę

```
Kod
<html>
<head>
  <script>
    title = 'The Beginning of the End';
    subTitle= 'Why I went to the Store';
    teaser = 'The day started like any other, but
    little did I know that it would soon turn
    into the worst day of my life!'
    dDate = '01'
    mDate = 'January'
    yDate = '2001'
  </script>
  <link rel="stylesheet"
href="default.css">
</head>
<body>
  <script src="header.js"></script>
  <!-- Begin Content -->
  <p>Et quid erat, quod me delectabat, nisi
  amare et amari?...</p>
  <!-- End Content -->
  <script src="footer.js"></script>
</body>
</html>
```



Rysunek 22.12. Końcowy wygląd strony z dynamicznie stworzoną stopką i nagłówkiem. Ponieważ kodu nagłówka i stopki nie umieszczamy na stronie, możemy zmienić jej układ w plikach JavaScript, a zmiany zostaną naniesione na każdej używającej ich stronie HTML

3. title="The Beginning of the End";

W znaczniku `<script>` w nagłówku dokumentu, w którym chcemy dodać nagłówki i stopkę, dodajmy zmienne, takie jak: `title`, `subTitle`, `teaser`, `dDate`, `mDate` i `yDate`. JavaScript wykorzysta te zmienne przy zapisywaniu nagłówka (kod 22.10).

4. `<script src="header.js"></script>`

W znaczniku `<body>` strony HTML dodaj znaczniki `<script>` z atrybutem źródła, którego wartość stanowi adres URL zewnętrznych plików JavaScript, zawierających nagłówki i stopkę (rysunek 22.12).

Wskazówki

- W nagłówku i stopce można umieścić dowolny kod HTML. Na przykład w mojej witrynie całą nawigację umieszczam w zewnętrznym pliku JavaScript — takim, jak te w tym przykładzie. Umożliwia mi to dodawanie i usuwanie elementów nawigacji bez konieczności zmieniania każdej strony w witrynie.
- Zmienne w punkcie 3. są tylko przykładem typu informacji, jakie można dodać do poszczególnych stron HTML, używając importowanego globalnego pliku JavaScript. W taki sposób można dołączyć każdy typ danych na temat artykułu, takich jak: rocznik, numer wydania lub jego położenie w witrynie.

Tworzenie własnych obramowań ramek

W używaniu ramek najbardziej denerwuje projektantów przestarzały wygląd obramowań standardowo umieszczanych pomiędzy ramkami (rysunek 22.13). Jednak używając właściwości tła, można stworzyć ramki w dowolnym stylu, jaki tylko możemy sobie wymarzyć (rysunek 22.14).

Chociaż takie obramowania mogą zostać umieszczone jedynie wzdłuż lewego lub górnego boku pojedynczej ramki, to i tak są bardzo pożyteczne przy zaznaczaniu granic pomiędzy poszczególnymi ramkami.

Aby stworzyć obramowanie ramki:

1. border.gif

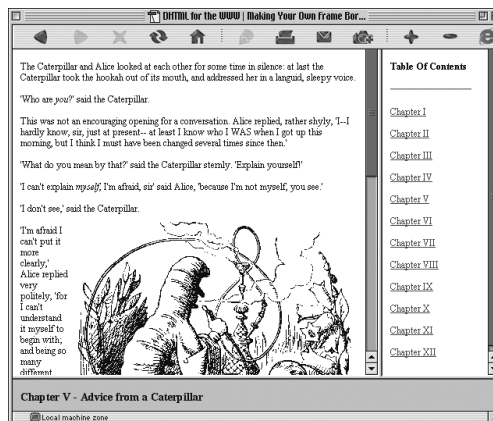
Stwórz grafikę, którą wykorzystasz do obramowania ramki. W tym przykładzie korzystam z ornamentu, który zapamiętałem jako `border.gif` (rysunek 22.15). Możecie użyć dowolnej grafiki, jaka się wam podoba.

2. index.html

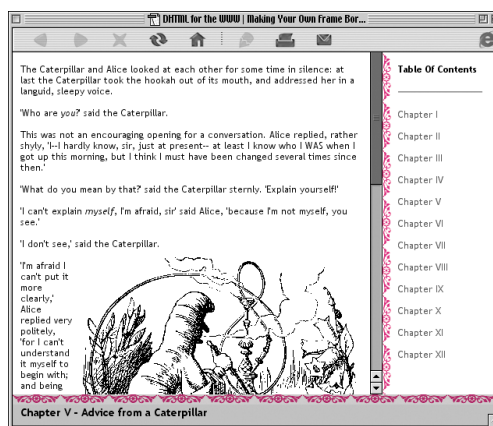
Stwórz dokument podzielony na ramki i upewnij się, że wyłączyłeś domyślne obramowanie:

```
border="0" framespacing="0"
frameborder="no"
```

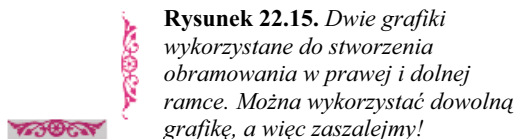
Zapisz ten zestaw ramek jako `index.html` (kod 22.11).



Rysunek 22.13. Dokument podzielony na ramki, z wyświetlonym domyślnym obramowaniem



Rysunek 22.14. Dokument podzielony na ramki ze stworzonym przy użyciu CSS rozdzielającym ramki obramowaniem w postaci czerwonego ornamentu



Rysunek 22.15. Dwie grafiki wykorzystane do stworzenia obramowania w prawej i dolnej ramce. Można wykorzystać dowolną grafikę, a więc zaszalejmy!

Kod 22.11. Dokument podzielony na ramki

```

Kod
<html>
<frameset rows="*,40" border="0"
framespacing="0" frameborder="no">
  <frameset cols="*,150" border="0"
framespacing="0" frameborder="no">
    <frame src="center_frame.html"
name="center" noresize>
    <frame src="right_frame.html"
name="right" noresize>
  </frameset>
  <frame src="bottom_frame.html"
name="bottom" noresize scrolling="no">
</frameset>
</html>

```

Kod 22.12. Ramka z pionowym obramowaniem

```

Kod
<html>
<head>
  <style type="text/css">
body {
  background: white url(border2.gif)
repeat-y;}
  </style>
</head>
<body>
  <h4>Table Of Contents</h4>
  <hr width="90%" align="left">
  <p><a href="#">Chapter I</a></p>
</body>
</html>

```

Kod 22.13. Ramka z poziomym obramowaniem

```

Kod
<html>
<head>
  <style type="text/css">
body {
  background: silver url(border1.gif)
repeat-x 0px 0;}
  </style>
</head>
<body>
  <h3>Chapter V - Advice from a
Caterpillar</h3>
</body>
</html>

```

3. right_frame.html

Wykorzystaj właściwość `background` w znaczniku `<body>` dokumentu HTML (kod 22.12 i 22.13), aby umieścić grafikę ornamentu stworzoną w punkcie 1. w tle odpowiednich ramek (patrz rysunek 22.14). Powiel grafikę albo w pionie (`repeat-y`), albo w poziomie (`repeat-x`) (zobacz *Ustawianie tła* w rozdziale 6.).

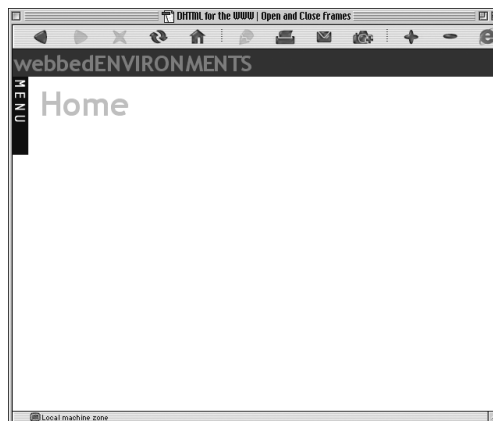
Wskazówki

- Technika sprawdza się tylko przy ustawianiu obramowania na górze ramki lub po jej lewej stronie. Nie można ustawić obramowań na obu brzegach jednocześnie, na dole lub po prawej stronie ramki.
- Wygląd obramowania może być dowolny i może być dowolnie duży. Należy tylko pamiętać, że obrazek powielany jest w określonym przez nas kierunku.
- Takie obramowania mają jedną ogromną wadę w porównaniu z domyślnym stylem obramowania. Ani autor, ani odwiedzający nie może użyć tych obramowań, aby zmienić wielkość ramki.

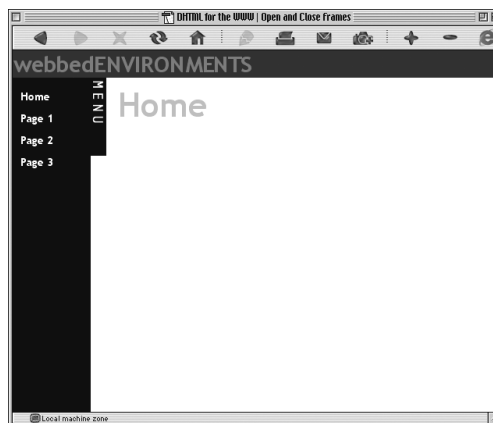
Otwieranie i zamykanie ramek

Wadą ramek jest to, że „monopolizują” obszar ekranu poprzez umieszczanie na nim na stałe tytułów i menu. Jeśli posiadamy duży monitor, może działać to znakomicie. Jednak osoby mające mniejsze monitory mogą wystraszyć się tym, co zobaczą.

Oto technika, którą opracowałem, wykorzystująca zagnieżdżone zestawy ramek oraz trochę skryptu JavaScript, aby otworzyć i zamknąć menu znajdujące się w ramce (rysunki 22.16 i 22.17). Gdy menu jest zamknięte, obszar zawartości okna może wykorzystać dowolną potrzebną przestrzeń.



Rysunek 22.16. Menu jest zamknięte



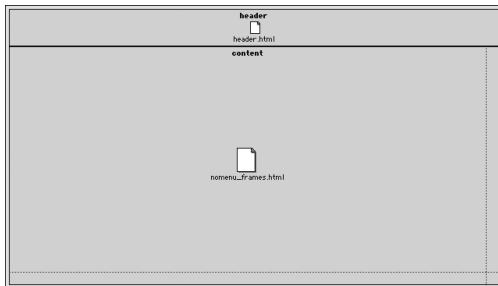
Rysunek 22.17. Menu jest otwarte

Kod 22.14. Dokument podzielony na ramki wykorzystane do stworzenia układu strony

```

<html>
<frameset rows="35,*" border="0"
framespacing="0" frameborder="no">
  <frame name="header" src="header.html"
scrolling="no" frameborder="no" noresize
marginHeight="2px" marginWidth="2px">
  <frame name="content"
src="nomenu_frames.html" marginwidth="10"
marginheight="10" frameborder="no">
</frameset>
</html>

```



Rysunek 22.18. Zestaw ramek pliku index.html

Kod 22.15. Zestaw ramek, który zostanie załadowany do ramki o nazwie content na stronie index.html, gdy menu jest zamknięte

```

<html>
<frameset cols="25,*" border="0"
framespacing="0" frameborder="no">
  <frame name="control" src="control.html"
marginwidth="0" marginheight="0"
scrolling="no" frameborder="no" noresize>
  <frame name="content2" src="content.html"
marginwidth="10" marginheight="10"
scrolling="auto" frameborder="no">
</frameset>
</html>

```

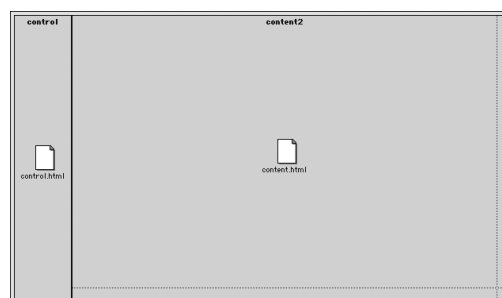
Aby stworzyć zamykające się ramki:

1. index.html

Stwórz główny zestaw ramek. W tym przykładzie stworzyłem dwie ramki: header i content (kod 22.14 i rysunek 22.18). Ramka o nazwie header będzie statyczna (nie zmienia swojego źródła) i będzie zawierać plik header.html z większością JavaScriptu, potrzebnego nam do wykonania tego efektu. W ramce content początkowo załadowana jest strona nomenu_frames.html, która także podzielona jest na ramki. Następnie źródło ramki zamieniane jest na stronę menu_frames.html, aby przełączyć wygląd na wersję z otwartym menu.

2. nomenu_frames.html

Stwórz zestaw ramek, dzielący stronę na dwie kolumny, i zapisz go jako nomenu_frames.html (kod 22.15 oraz rysunek 22.19). Pierwsza ramka o nazwie control zawiera plik control.html, druga — nazwana content2 — początkowo zawiera stronę content.html.



Rysunek 22.19. Podzielona na ramki strona nomenu_frames.html zostanie załadowana do ramki o nazwie content na stronie index.html wtedy, gdy menu jest zamknięte

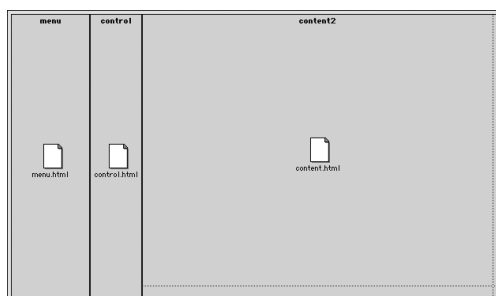
Rozdział 22.

3. menu_frames.html

Stwórz zestaw ramek, dzielący stronę na trzy kolumny, i zapamiętaj go jako menu_frames.html (kod 22.16 i rysunek 22.20). Druga i trzecia ramka w tym zestawie są identyczne jak zestaw ramek z pliku *nomenu_frames.html*. Pierwsza ramka ma szerokość 150 pikseli i wyświetla plik menu.html. Odwiedzający będą mogli przełączać się pomiędzy dwoma zestawami ramek, utworzonymi w punkcie 2. i 3., w zależności od tego, czy będą chcieli, by menu było otwarte, czy nie.

Kod 22.16. Podzielony na ramki dokument ładowany jest do ramki content pliku index.html, gdy menu jest otwarte

```
Kod
<html>
<frameset cols="100,25,*" border="0"
framespacing="0" frameborder="no">
  <frame name="menu" src="menu.html"
marginwidth="10" marginheight="10"
scrolling="no" frameborder="no" noresize>
  <frame name="control" src="control.html"
marginwidth="0" marginheight="0"
scrolling="no" frameborder="no" noresize>
  <frame name="content2" src="content.html"
marginwidth="10" marginheight="10"
scrolling="auto" frameborder="no">
</frameset>
</HTML>
```



Rysunek 22.20. Podzielona na ramki strona menu_frames.html ładowana jest do ramki content na stronie index.html, gdy menu jest otwarte

Kod 22.17. Dokument HTML wykorzystany w ramce header na stronie *index.html*. Zawiera funkcję *menuToggle()*. Gdy funkcja zostanie uruchomiona, zamyka lub otwiera ramkę menu

```

Kod
<html>
  <head>
    <script language="JavaScript">
      var frameState = 0;
      var contentSRC = null;
      function menuToggle() {
        if (frameState == 0) {
          contentSRC =
parent.content.content2.location.href;
          top.content.location =
'menu_frames.html';
          frameState = 1;
          return;
        }
        else {
          contentSRC =
parent.content.content2.location.href;
          top.content.location =
'nomenu_frames.html';
          frameState = 0;
          return;
        }
      }
    </script>
    <style type="text/css">
      h2 {color: gray; font-weight: bold; font-
size: 24px; font-family: "Trebuchet MS",
Arial, Helvetica, Geneva, sans-serif;}
      body {background-color: #333; }
    </style>
  </head>
  <body>
    <h2>webbedENVIRONMENTS</h2>
  </body>
</html>

```

4. header.html

Stwórz plik HTML, który będzie wykorzystany w ramce o nazwie header w pliku stworzonym w punkcie 1. i zapamiętaj go jako *header.html* (kod 22.17). Funkcja *menuToggle()* jest kluczową funkcją na tej stronie. Wykonywana jest wtedy, gdy odwiedzający kliknie odnośnik „Menu” w pliku *control.html*. Funkcja najpierw sprawdza, jaki dokument jest załadowany do ramki *content2* i przechowuje ten URL jako zmienną *contentSRC*. Następnie sprawdza, czy menu jest widoczne i zmienia zestaw ramek na *nomenu_frames.html* albo na *menu_frames.html*. Zmienna *frameState* zapisuje stan menu: 0 (zamknięte) lub 1 (otwarte).

Rozdział 22.

5. menu.html

Stwórz plik HTML, który będzie wyświetlany wtedy, gdy menu będzie otwarte i zapisz go jako menu.html (kod 22.18). Ja w przykładzie użyłem odnośników HTML, jednak w pliku tym można umieścić dowolną zawartość HTML. Wszystkim odnośnikom powinno się przypisać ramkę, w której mają zostać wyświetlone (target="content2").

6. control.html

Stwórz plik HTML, który zostanie wykorzystany w ramce control i zapisz go jako control.html (kod 22.19). Plik zawiera odnośnik, który — gdy zostanie kliknięty — uruchamia funkcję menuToggle() z punktu 4.

Kod 22.18. Dokument HTML zawierający menu, które będzie wyświetlane w ramce o nazwie menu w pliku menu_frames.html. Dokument może zawierać dowolną zawartość HTML

```
Kod
<html>
<head>
  <style type="text/css">
    body {color: gray; font: bold 12px/24px
      "Trebuchet MS", Arial, Helvetica, Geneva,
      sans-serif; background-color: black;}
    a:link {color: white; text-decoration: none;
    }
    a:hover {color: white; text-decoration:
      underline; }
    a:active {color: silver; text-decoration:
      underline; }
    a:visited {color: white; text-decoration:
      none; }
  </style>
</head>
<body>
  <a href="home.html"
target="content2">Home</a><br>
  <a href="page1.html"
target="content2">Page 1</a><br>
  <a href="page2.html"
target="content2">Page 2</a><br>
  <a href="page3.html"
target="content2">Page 3</a><br>
</body>
</html>
```

Kod 22.19. Dokument HTML wykorzystany w ramce o nazwie control w pliku menu_frames.html i nomeniu_frames.html. Plik zawiera odnośnik, uruchamiający funkcję menuToggle(), umieszczoną w pliku header.html

```
Kod
<html>
<body>
  <div id="tabs">
    <a
href="javascript:top.header.menuToggle();">
      
    </a>
  </div>
</body>
</html>
```

Kod 22.20. Plik jest początkowo wyświetlany w ramce content2 w plikach menu_frames.html i nomeni_frames.html. JavaScript, znajdujący się w pliku, sprawdza adres URL strony, która była wyświetlona, zanim menu zostało zmienione (zapisany w zmiennej contentSRC w pliku header.html) i ładuje stronę w tę ramkę

```
Kod
<html>
<head>
  <script language="JavaScript">
    var contentSRC;
    function replaceContent() {
      contentSRC = top.header.contentSRC;
      if (contentSRC == null) {self.location =
        'home.html';
      } else self.location = contentSRC;
    }
  </script>
</head>
<body onload="replaceContent();">
</body>
</html>
```

7. content.html

Stwórz stronę HTML, zawierającą kod 22.20. Strona ta jest krokiem pośrednim i nigdy nie jest zbyt długo widoczna na ekranie. Kontroluje tylko, jaki dokument był załadowany do ramki content2 poprzez sprawdzenie wartości przypisanej zmiennej contentSRC zapisanej w ramce header, i załaduje ten dokument. Jeśli wcześniej nie był w niej wyświetlony żaden plik (na przykład strona jest ładowana po raz pierwszy), to ładowany jest plik home.html.

8. home.html

Stwórz wszystkie strony dla witryny (kod 22.21). Wszystkie będą wyświetlane w ramce content2.

Wskazówki

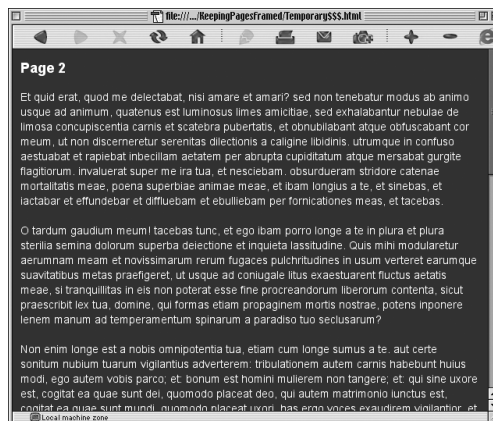
- Technika umożliwia odwiedzającemu stronę otwarcie i zamknięcie menu nawigacyjnego, bez „zgubienia” miejsca, w którym znajdował się w witrynie.
- W zamykanej ramce nie musi znajdować się menu. Tę technikę zastosowałem w intranecie pewnej organizacji do stworzenia kalendarza zdarzeń, który mógł być otwierany w dowolnym momencie.
- Ze względu na pewne ograniczenia związane z bezpieczeństwem, w Internet Explorerze i Netscape Navigatorze nie można otwierać lub zamykać menu, gdy ramka content2 zawiera dokument spoza serwera, na którym znajduje się witryna, a więc nie można wykorzystywać zawartości z innych witryn.

Zachowanie podziału strony na ramki

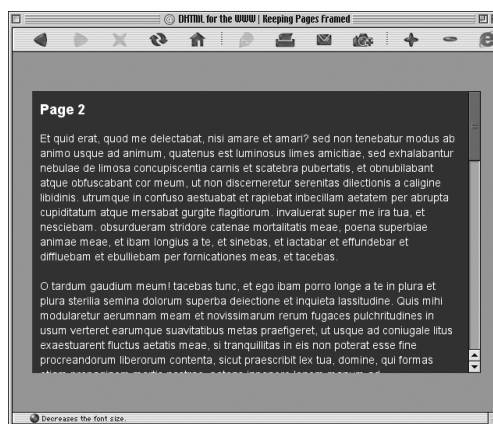
Wyobraźmy sobie, że czytamy książkę. Wieczorem, kończąc czytać, wkładamy zakładkę na stronę, którą przeczytaliśmy jako ostatnią i odkładamy książkę. Następnego wieczoru podnosimy książkę i chcemy rozpocząć czytanie w miejscu, w którym poprzednio skończyliśmy. Jednak przez jakieś dziwne okoliczności zakładka w rzeczywistości znalazła się z powrotem za okładką, a my musimy przekartkować wcześniej przeczytane strony, aby znaleźć miejsce, w którym skończyliśmy. Jeśli tworzymy stronę przy użyciu ramek, to takie sytuacje mogą spotkać również naszych odwiedzających. Chcą umieścić zakładkę na jednej z wewnętrznych stron, ale okazuje się, że zakładka znalazła się za okładką.

Powyższa sytuacja nie jest jedynym problemem, związanym z wywoływaniem stron podzielonych na ramki. Co się stanie, jeśli zechcemy w e-mailu umieścić odnośnik do konkretnej strony wewnątrz witryny lub stworzyć taki odnośnik w innej witrynie? Pewnie, można wysłać URL pojedynczej strony, ale jeśli wyślemy odwiedzających do tej konkretnej strony bez ramek, to będzie tak, jakbyśmy dali im książkę bez pozostałych stron lub bez grzbietu. Można by stworzyć dokument podzielony na ramki dla każdej ze stron osobno, ale byłoby to problematyczne i nieporęczne.

Jest łatwiejszy sposób. Co prawda nie możemy zmienić sposobu, w jaki działają ramki i z pewnych powodów twórcy przeglądarek zignorowali te poważne problemy ich używalności, ale możemy w naszej witrynie zaimplementować specjalne obejście tego problemu i pomóc naszym odwiedzającym (rysunki 22.21 i 22.22).



Rysunek 22.21. Strona początkowo ładuje się normalnie, ale prawie natychmiast...



Rysunek 22.22. ...ekran mruga i strona, którą oglądaliśmy, znajduje się już wewnątrz odpowiedniego zestawu ramek

Kod 22.22. Ten mały zewnętrzny plik JavaScript sprawia, że strona znajdująca się w ramkach będzie znajdować się w nich nadal. Jak tylko załaduje się strona zawierająca ten kod, to zostanie przeładowana i wyświetlona w odpowiedniej formie strony podzielonej na ramki

```
Kod
myPage = self.location;
thisPage = top.location;

if (thisPage == myPage) {
    contentSRC = escape(myPage);
    frameURL = 'index.html?' + contentSRC;
    top.location.href = frameURL;
}
```

Kod 22.23. Zestaw ramek nie jest kodowany bezpośrednio w HTML-u, ale jest kodowany przy użyciu metody JavaScript `document.write`, tak byśmy mogli dodać zmienne do HTML-a

```
Kod
<html>
<script language="JavaScript">
    contentSRC =
    (location.search.substring(1) ?
    location.search.substring(1) :
    'defaultContent.html');
    contentSRC = unescape(contentSRC);
    var writeFrame = ''
    writeFrame += '<FRAMESET COLS="*,575,*"
    BORDER="0" FRAMESPACING="0"
    FRAMEBORDER="NO">';
    writeFrame += '<FRAME
    SRC="filler.html">';
    writeFrame += '<FRAMESET
    ROWS="50,*,50">';
    writeFrame += '<FRAME
    SRC="filler.html">';
    writeFrame += '<FRAME SRC="" + (
    contentSRC ) + " NAME="content" NORESIZE>';
    writeFrame += '<FRAME
    SRC="filler.html">';
    writeFrame += '</FRAMESET>';
    writeFrame += '<FRAME
    SRC="filler.html">';
    writeFrame += '</FRAMESET>';
    document.write(writeFrame);
</script>
</html>
```

Aby automatycznie umieścić stronę HTML wewnątrz odpowiedniego zestawu ramek:

1. framed.js

Stwórz zewnętrzny plik JavaScript (zobacz *Używanie zewnętrznych plików JavaScript* w rozdziale 23.) i zapisz go jako `framed.js` (kod 22.22). Dodaj skrypt, który najpierw sprawdza, czy dokument jest załadowany do strony podzielonej na ramki (porównując adres URL strony z adresem URL całego okna). Jeśli dokument jest załadowany do ramki, to adresy są różne i nic się nie dzieje. Jeśli adresy są takie same (strony nie otacza żaden zestaw ramek), to skrypt łączy URL strony podzielonej na ramki, w której powinna znajdować się strona (w tym przypadku `index.html`), z adresem URL samej strony (zmienna `myPage`), oddzielając te dwa adresy URL od siebie za pomocą znaku zapytania (?). Funkcja kończy swoje działanie, zmieniając stronę na nową, której adres podany jest w zmiennej `frameURL`.

2. index.html

Stwórz dokument podzielony na ramki, użyj JavaScriptu, aby zapisać kod HTML na stronie i zapamiętaj plik jako `index.html` (kod 22.23). JavaScript wydobywa adres URL strony, która ma być załadowana do ramki `content` z adresu URL strony, która jest aktualnie załadowana. Jeśli znak zapytania (?) pojawia się w głównym adresie strony, to skrypt używa części URL, znajdującej się za znakiem zapytania, jako źródła ramki `content` zapisanej w zmiennej `contentSRC`. W przeciwnym razie skrypt wykorzystuje domyślny URL, w tym przypadku jest to plik `defaultContent.html`, ale możemy w tym miejscu wykorzystać dowolny plik.

3. content.html

Stwórz strony, które będą wyświetlane w witrynie (kody 22.24 i 22.25). Podłącz zewnętrzny plik JavaScript stworzony w punkcie 1. w znaczniku `<head>` wszystkich dokumentów, które chcesz dynamicznie umieszczać w ramce content w zestawie ramek:

```
<script src="framed.js"></script>
```

Wskazówki

- Jeśli chcemy skierować kogoś bezpośrednio do jakiejś strony wewnątrz naszej witryny, wystarczy wysłać tej osobie URL do dokumentu stworzonego w punkcie 2. (index.html). Strona automatycznie umieści się w odpowiedniej ramce, jeśli tylko będzie do niej zaimportowany plik framed.js, jak pokazano w punkcie 3.
- Gdy odwiedzający podaży za odnośnikiem do jednej z tych stron i umieści tam zakładkę, to powróci do tej strony, znajdującej się wewnątrz otoczenia pozostałych ramek.
- Netscape Navigator dla Windows i Internet Explorer dla MacOS i Windows umożliwiają założenie strony w zestawie ramek poprzez kliknięcie i przytrzymanie przycisku (MacOS) lub naciśnięcie prawego przycisku myszy (Windows) i wybranie odpowiedniej opcji z wyskakującego menu. Jeśli odwiedzający założy stronę wewnątrz struktury ramek stworzonej przy użyciu pokazanej tu techniki, to — gdy powróci do strony przez zakładkę — strona załaduje się do ramek automatycznie.
- Technika nie rozwiązuje wszystkich problemów związanych z zakładaniem stron w ramach, ale dopóki twórcy przeglądarek nie skoordynują swoich wysiłków, niewątpliwie pomoże zwiększyć użyteczność stron w ramach.

Kod 22.24. Ta strona jest ładowana, jeśli w adresie URL nie znajduje się znak zapytania (?). Do strony zaimportowany jest zewnętrzny plik framed.js, wymuszający w razie konieczności umieszczenie strony w ramach

```
Kod
<html>
<head>
  <script src="framed.js"></script>
  <link rel="stylesheet"
href="default.css">
</head>
  <body>
    <h2>Page 1</h2>
  </body>
</html>
```

Kod 22.25. Strona zawiera podłączenie do kodu framed.js

```
Kod
<html>
<head>
  <script src="framed.js"></script>
  <link rel="stylesheet"
href="default.css">
</head>
  <body>
    <h2>Page 2</h2>
  </body>
</html>
```

Dobry wygląd w druku (w Sieci)

Nie spotkałem jeszcze nigdy biura, w którym w ogóle nie korzystano by z papieru i byłbym zaskoczony, gdybym do takiego trafił. Ale wraz z nadejściem komputerów przyszła obietnica wyeliminowania papieru z naszego życia — koniec z zapchanymi i zagrożonymi szafkami, koniec wycinania drzew — tylko wolna od entropii utopia, w której są bez przerwy utylizowane i ponownie wykorzystywane elektrony, dokładnie tak jak w „Star Treku”.

Ale coś mi mówi, że zanim uda się nam wyeliminować z naszego życia papier, odkryjemy technologie, pozwalające podróżować do najbardziej oddalonych gwiazd.

Wraz z pojawieniem się drukarek laserowych i atramentowych zaczęliśmy zakopywać się w hałdach doskonałych wydruków. Wydaje się, że Sieć wpływa nawet na zwiększenie liczby robionych przez nas wydruków. Jeśli strona jest dłuższa niż parę przewinieć, większość osób drukuje ją.

Ale Internet został stworzony, aby wyświetlać informacje na ekranie, a nie na papierze. Grafiki używane w Sieci wyglądają w druku „klockowato”, a zwykłemu HTML-owi brakuje wiele do dobrej kontroli układu. Możemy jednak poczynić kilka kroków, aby poprawić wygląd drukowanych stron. Dobry wygląd strony po wydrukowaniu może kosztować nas trochę wysiłku, ale odbiorcy będą nam za to wdzięczni.

Rozdział 22.

Oto osiem prostych zasad, które możemy wprowadzić w życie, chcąc poprawić wygląd naszych stron po wydrukowaniu:

- *Używajmy CSS.* Kaskadowe arkusze stylów są narzędziem projektowania internetowego. CSS umożliwia tworzenie dokumentów wyglądających w druku tak samo dobrze, jak te tworzone przy użyciu edytora tekstu.
- *Określajmy media.* CSS pozwala zdefiniować różne arkusze stylów, które będą stosowane w zależności od tego, w jaki sposób będzie pokazywana strona — czy to na ekranie, czy na papierze (zobacz *Tworzenie arkusza stylów dla wydruków* w rozdziale 2.).
- *Używajmy podziałów strony, aby nagłówki były drukowane wraz z tekstem, który poprzedzają.* Choć atrybut podziału strony nie jest jeszcze szeroko obsługiwany (zobacz *Ustawianie podziału strony przy drukowaniu* w rozdziale 4.), to już wkrótce może się stać uniwersalnym standardem.
- *Korzystajmy z ramek, aby oddzielić nawigację od zawartości.* Spróbujmy umieszczać główną treść (to, co chcą przeczytać nasi odbiorcy) w osobnej ramce. Można wtedy umieścić nawigację, tytuły i ogłoszenia w różnych ramkach. Umożliwi to odwiedzającym wydrukowanie tylko ramki zawierającej to, co ich interesuje i nieprzejmowanie się resztą (zobacz *Otwieranie i zamykanie ramek wcześniej* w tym rozdziale).
- *Unikajmy korzystania z kolorów lub grafik tła z jasnym tekstem.* Choć mogą one dodać naszej stronie pewnego smaku, wyglądają fatalnie na wydruku. Niektóre przeglądarki umożliwiają drukowanie dokumentów bez tła, ale jasny tekst nie będzie widoczny na białym tle papieru.
- *Unikajmy korzystania z przezroczystych kolorów w grafikach,* zwłaszcza jeśli grafika znajduje się na tle innej grafiki lub koloru tła innego niż biały. Przezroczysty obszar w rysunkach GIF zazwyczaj drukowany jest jako biały, bez względu na to, jaki kolor znajdował się za nim w oknie. Taka sytuacja nie sprawi kłopotu, jeśli grafika znajduje się na białym tle, ale efekty wypadną koszmarnie, jeśli grafika znajdzie się na ciemnym tle.
- *Unikajmy używania tekstu w grafice.* Ironia drukowania materiałów z Sieci polega na tym, że tekst w grafice — wyglądający dobrze w oknie — wygląda strasznie „klockowato” na papierze, ale zwyczajny tekst HTML, który może wyglądać „klockowato” na ekranie, drukuje się dobrze na każdej przyzwoitej drukarce. Starajmy się używać tekstu HTML wszędzie tam, gdzie to tylko możliwe.
- *Umieśćmy osobną, gotową do druku wersję dokumentu.* Zamiast zmuszać odwiedzających, aby podążali za każdym odnośnikiem w naszej witrynie i drukowali każdą stronę osobno, umieśćmy podobny dokument, który odwiedzający mogą pobrać i wydrukować. Zastosowanie Adobe Acrobat jest dobrym sposobem na udostępnienie mniej więcej uniwersalnego formatu plików, który przy dostarczaniu dokumentów przez sieć zachowuje większość stylów formatowania, czcionki i grafiki. Więcej na temat Acrobat można dowiedzieć się na stronach internetowych Adobe (www.adobe.com).