Kuloodporne strony internetowe

Jak poprawić elastyczność z wykorzystaniem XHTML-a i CSS

WYDANIE III

Dan Cederholm

AUTORYTETY INFORMATYKI

New Rider

Helion

Tytuł oryginału: Bulletproof Web Design: Improving flexibility and protecting against worst-case scenarios with HTML5 and CSS3 (3rd Edition)

Tłumaczenie: Piotr Cieślak

ISBN: 978-83-246-5120-7

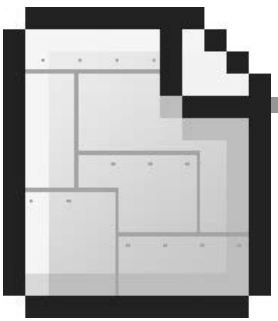
Authorized translation from the English language edition, entitled:
BULLETPROOF WEB DESIGN: IMPROVING FLEXIBILITY AND PROTECTING AGAINST WORST-CASE SCENARIOS WITH HTML5 AND CSS3, Third Edition;
ISBN 0321808355; by Daniel Cederholm; published by Pearson Education, Inc, publishing as New Riders Publishing.

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/kuloo3
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści



3

ELASTYCZNE WIERSZE

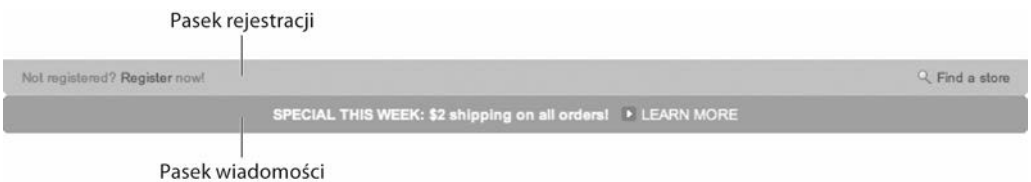
Nie należy definiować wysokości poziomych elementów strony i trzeba zaplanować możliwość powiększenia ich w pionie.

Poziome elementy strony, takie jak nagłówki, paski logowania, ścieżki odnośników do odwiedzonych stron oraz paski wyszukiwania, to elementy często spotykane w typowej witrynie internetowej. Są one zwykle umieszczone na górze strony i mogą zawierać różne elementy graficzne (na przykład tło lub obrazki) oraz tekst. Obszary te często są zaprojektowane tak, aby uniemożliwić *rozciągnięcie ich w pionie* — przy tworzeniu takich projektów zazwyczaj zakłada się, że tekst nie będzie powiększany, a w elementach nie będzie umieszczona zbyt duża ilość treści, lub że takie działania nie zmienią wyglądu strony. Zwyczajowo elementy, w których umieszczane są całe artykuły lub duże fragmenty tekstu, dopasowują się do dowolnej długości i wielkości tekstu lub innych zawartych w nich treści. Ważne jest — i nie jest to niemożliwe — by inne poziome elementy strony także traktować w ten sposób.

W tym rozdziale zajmiemy się częstym rozwiązaniem stosowanym przy tworzeniu obszarów na reklamy lub miejsc logowania, które umieszczane są w górnej części strony internetowej. Zajrzymy takiemu typowemu projektowi „pod maskę", a następnie zrekonstruujemy go w taki sposób, aby pomieścił tekst dowolnego rozmiaru i każdą jego ilość.

Typowe rozwiązanie

Rozciągnięcie w pionie poziomego elementu strony wyjaśnimy na przykładzie wziętym z internetu — z witryny pewnego sklepu internetowego… nazwijmy go „Najlepszy sklep wszech czasów" (oczywiście sklep ten w rzeczywistości tak się nie nazywa, zmieniliśmy jego nazwę, by zachować anonimowość winowaj… znaczy, autorów). „Najlepszy sklep wszech czasów" to typowy sklep internetowy, w którym oferuje się wiele mniej lub bardziej przydatnych produktów dla domu. W przykładzie pokazaliśmy fikcyjną stronę, ale wykorzystane techniki są powszechnie stosowane przy tworzeniu projektów tego typu. W prezentacji typowego rozwiązania ograniczymy się tylko do poziomych pasków logowania i promocji, zastosowanych w witrynie „Najlepszego sklepu wszech czasów" (rysunek 3.1). Na górze każdej strony znajduje się pasek umożliwiający zalogowanie się oraz odszukanie najbliższego sklepu, a zaraz pod nim pasek z regularnie aktualizowanymi informacjami dotyczącymi promocji. Każdy z nich zawiera tylko jedną linię tekstu.



Rysunek 3.1. Górna część strony przykładowego „Najlepszego sklepu wszech czasów"

Widoczne powyżej paski (oraz cały układ strony) zbudowane są za pomocą wielu zagnieżdżonych tabel. Elementy graficzne (takie jak zaokrąglone końce każdego paska) oraz tekst umieszczone są w osobnych komórkach.

Na rysunku 3.2 widoczny jest prawdopodobny sposób podzielenia pasków na osobne komórki — ich granice zostały zaznaczone czerwoną linią. Każdy segment paska, z obrazkami w postaci zaokrąglonych narożników włącznie, został umieszczony w osobnej komórce. Jest to tylko *przybliżenie* prawdziwej struktury tabel źródłowego projektu, gdyż nie chcieliśmy za bardzo wnikać w jego strukturę. Najważniejszy wniosek jest taki, że do utworzenia widocznego na rysunkach projektu posłużyły tabele, wypełniacze GIF i małe grafiki.

Rysunek 3.2. Każda część paska znajduje się w osobnej komórce tabeli

Wykorzystywanie tabel i wypełniaczy GIF do pozycjonowania grafiki oraz tekstu jest techniką, która przez lata stosowania została udoskonalona do perfekcji. Większość stron internetowych została zbudowana w taki sposób, a wielu projektantów szczyciło się tym, że potrafili co do piksela odtworzyć na stronie internetowej dowolny projekt graficzny. Postępowano w myśl zasady, że jeśli coś da się zaprojektować i *wydrukować*, to można to przekształcić na stronę internetową.

Powstają jednak coraz lepsze techniki tworzenia stron internetowych. Odkrywamy metody, które zwiększają czytelność stron oraz ich dostępność przy wykorzystaniu zwięzłego, poprawnego semantycznie języka znaczników oraz stylów CSS. W dalszej części tego rozdziału zastosujemy te metody do przebudowania dwóch poziomych pasków w witrynie „Najlepszego sklepu wszech czasów". Jednak najpierw pokażemy, dlaczego dotychczasowy projekt nie jest kuloodporny.

Dlaczego rozwiązanie to nie jest kuloodporne

Do konstrukcji poziomych pasków można wprowadzić wiele ulepszeń, które poprawią elastyczność tego elementu strony. Przeanalizujmy wady przedstawionego rozwiązania, aby lepiej oszacować zakres zmian niezbędnych do przekształcenia go w prawdziwie kuloodporny projekt.

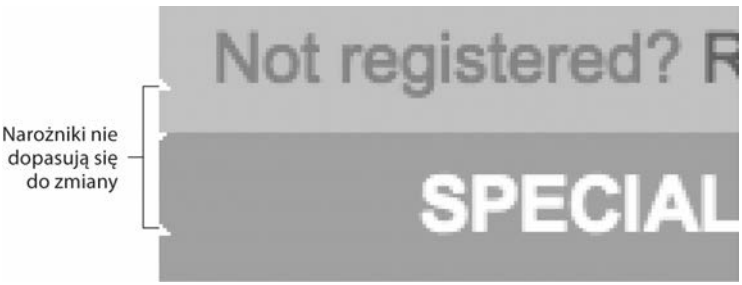
MAŁO ISTOTNE ELEMENTY GRAFICZNE

Jak już wspomnieliśmy wcześniej, oba paski są zbudowane przy użyciu wielu zagnieżdżonych tabel. Każda część paska znajduje się w osobnej komórce tabeli. **Mało istotne elementy graficzne**, takie jak zaokrąglone końce pasków, są umieszczone w kodzie wraz z tekstem. Takie elementy będą tylko przeszkadzać wszystkim użytkownikom wyświetlającym stronę w przeglądarce tekstowej lub odczytującym jej zawartość przy użyciu czytnika ekranu. Z pewnością pomogłoby zadeklarowanie odpowiednich wartości atrybutów `alt` dla wszystkich rysunków (z uwzględnieniem pustych deklaracji `alt=""` dla grafik pełniących rolę ozdobników), jednak projektanci pracujący przy tworzeniu witryny „Najlepszego sklepu wszech czasów" postanowili z tego zrezygnować.

Pojęcie „mało istotnej grafiki" odnosi się do elementów graficznych, które nie wnoszą niczego do funkcjonalności i treści witryny oraz nie pełnią żadnej istotnej roli informacyjnej. Zaokrąglone narożniki pasków to **element dekoracyjny**, dlatego w naszym rozwiązaniu przeniesiemy go z kodu strony do arkusza stylów.

STAŁE WYSOKOŚCI

Jeżeli spróbujemy odrobinę zwiększyć rozmiar tekstu, projekt poziomych pasków całkowicie się rozpadnie (rysunek 3.3). Zwiększenie wielkości tekstu to jeden z najlepszych testów na to, czy dany projekt jest elastyczny. Taki test umożliwia nie tylko sprawdzenie czytelności strony przy dużych wielkościach czcionek — dzięki niemu wiemy także, czy projekt poradzi sobie z prawidłowym wyświetleniem różnych ilości tekstu (niezależnie od jego rozmiaru). Załóżmy, że redaktor będzie chciał później wyświetlić *dwie* linie tekstu promocyjnego, a nie *jedną*, jak jest to zdefiniowane w pierwotnej specyfikacji. W kuloodpornej witrynie pasek powinien się *powiększyć* i dopasować do nowej ilości tekstu bez konieczności wprowadzania dodatkowych zmian. Takie rozwiązanie pozwala zaoszczędzić czas i pieniądze, a to powinno być wystarczającym powodem do jego stosowania.



Rysunek 3.3. Przy powiększaniu tekstu rysunek zaokrąglonych narożników nie rozciągnie się wraz z całym projektem

Przyglądając się rysunkowi 3.3, możemy stwierdzić, że obrazki dające efekt zaokrąglonych krawędzi pasków zostały zaprojektowane dla ich stałej, ściśle określonej wysokości. W rezultacie wstawienie do paska dowolnego elementu wyższego niż wysokość obrazka z narożnikami popsuje wygląd projektu. W rozdziale tym (oraz pozostałych) staram się uświadomić twórcom stron, że o *wysokości* elementów należy zacząć myśleć w diametralnie inny sposób. Możliwe jest tworzenie projektów, które będą dopasowywać się do większej czcionki, większej ilości tekstu oraz dowolnych elementów umieszczonych w ich wnętrzu. Jak to osiągnąć, pokażę za chwilę.

PRZEROŚNIĘTY KOD

Stosowanie tradycyjnych metod projektowania przy tworzeniu przykładowych poziomych pasków wymaga od projektanta napisania bardzo dużej ilości kodu. Jak pamiętamy z rozdziału 2., „Skalowalna

nawigacja", zagnieżdżanie tabel znacznie zwiększa ilość kodu tworzącego stronę. Taki przerost kodu nad treścią zatyka serwery, zmniejsza przepustowość łącza i sprawia, że strona może być zupełnie niezrozumiała dla programów niebędących przeglądarkami oraz niektórych urządzeń. Na szczęście istnieje znacznie prostsza, elastyczna metoda, która pozwala uzyskać ten sam efekt wizualny. Zapoznajmy się z nią.

Rozwiązanie kuloodporne

Podczas tworzenia kuloodpornego projektu poziomych pasków najpierw skupimy się na strukturze kodu. Później zdefiniujemy kolory, rozmieścimy poszczególne elementy oraz obrazki tła. Pod koniec rozdziału uzyskamy elastyczny komponent, który będzie się dopasowywał do każdego rozmiaru tekstu i do każdej jego ilości. Zacznijmy więc od struktury kodu i zmniejszmy jej objętość.

STRUKTURA KODU

Wyboru struktury pasków możemy dokonać tylko na podstawie analizy ich zawartości. Jakie elementy najlepiej sprawdzą się w tej roli? Jakie elementy najlepiej opiszą zawartość pasków? Gdy zaczynamy tworzyć rozwiązanie od samego początku, najlepiej jest najpierw odpowiedzieć sobie na pytania tego typu i na tej podstawie wybrać odpowiednie elementy. Oczywiście może istnieć kilka poprawnych odpowiedzi, jednak pytania te i tak należy sobie zadać przed napisaniem pierwszej linijki kodu.

W tym przypadku potrzebujemy dwóch elementów pełniących rolę pojemników, po jednym dla każdego paska. Fragmenty tekstu umieszczone na obydwu końcach pierwszego paska widzę jako pozycje listy, natomiast w drugim pasku znajduje się po prostu jeden akapit tekstu.

Napiszmy więc cały kod potrzebny do utworzenia naszego komponentu:

```
<ul>
    <li>Not registered? <a
href="/register/">Register</a>now!</li>
    <li><a href="/find/">Find a store</a></li>
</ul>
<div>
    <p><strong>Special this week:</strong> $2 shipping
on all orders! <a href="/special/">LEARN MORE</a></p>
</div>
```

Górny pasek przedstawiliśmy jako listę składającą się z dwóch elementów, a dolny jako jeden akapit zawarty wewnątrz elementu . Projekt wygląda teraz ładnie, prosto i bardzo schludnie — pozbyliśmy się przerośniętego kodu z zagnieżdżonymi tabelami.

Osiągnęliśmy także nasz cel w zakresie poprawienia dostępności treści. Bez względu na to, jakie urządzenie bądź oprogramowanie zostanie użyte do wyświetlenia strony, te dwa paski zawsze zostaną zinterpretowane jako lista i jeden akapit, czyli struktura odpowiadająca umieszczonej w niej treści.

IDENTYFIKACJA ELEMENTÓW

Naszym kolejnym zadaniem jest unikatowe oznaczenie wszystkich elementów, do których będziemy chcieli zastosować style. Przypisanie kilku elementom identyfikatorów (atrybut id) umożliwi nam późniejsze rozmieszczenie i pokolorowanie składników projektu oraz dodanie obrazków — czyli przekształcenie prostego kodu w gotowy projekt.

```
<ul id="register">
    <li id="reg">Not registered? <a
href="/register/">Register</a> now!</li>
    <li id="find"><a href="/find/">Find a store</a></li>
</ul>
<div id="message">
    <p><strong>Special this week:</strong> $2 shipping
on all orders! <a href="/special/">LEARN MORE</a></p>
</div>
```

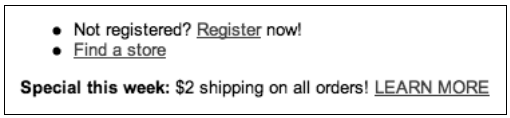
Właśnie dodaliśmy unikatowe atrybuty id do listy, jej dwóch elementów oraz do elementu zawierającego drugi pasek. Identyfikatory te można sobie wyobrazić jako **uchwyty**, do których już za chwilę dołączymy style. Można się zastanawiać, dlaczego identyfikator #message nie został przyporządkowany do samego elementu <p>, co wyeliminowałoby konieczność wstawiania otaczającego go elementu . Do skończenia projektu potrzebne nam jednak będą oba elementy, a dodatkowy element pełniący rolę kontenera tu i tam jest często niezbędny (jego zastosowanie nie jest przestępstwem!), kiedy tworzy się elastyczne, ale rozbudowane projekty stron internetowych. Byle tylko z umiarem!

UWAGA

Zanim zaczniemy przypisywać style, przyjmujemy, że cały projekt ma zdefiniowaną konkretną szerokość (w przypadku witryny „Najlepszego sklepu wszech czasów" wynosi ona 768 pikseli). Innymi słowy, zakładamy, że tworzone paski znajdują się wewnątrz jakiegoś kontenera (na przykład `<div>` lub `<table>`) o ściśle określonej szerokości.

BEZ STYLÓW

Na rysunku 3.4 widzimy zdefiniowaną wcześniej strukturę po wyświetleniu w przeglądarce internetowej. Arkusz stylów zawiera na razie tylko podstawowe deklaracje związane z wyglądem tekstu (w witrynie „Najlepszego sklepu wszech czasów" używana była czcionka Arial).



Rysunek 3.4. Podgląd kuloodpornej struktury kodu (efekt uzyskany przed zastosowaniem stylów). Taki projekt jest czytelny i zrozumiały dla każdego urządzenia

Na urządzeniu, które nie obsługuje CSS, strona wyświetlona zostanie tak, jak na rysunku 3.4. Taki szkielet jest nadal czytelny oraz zrozumiały dla każdego urządzenia odtwarzającego stronę. Teraz możemy przystąpić do definiowania stylów.

W arkuszu stylów utworzyliśmy deklarację dla elementu `<body>`, w której przy użyciu słowa kluczowego `small` ustaliliśmy podstawowy rozmiar tekstu.

```
body {
    font-family: Arial, sans-serif;
    font-size: small;
}
```

WSKAZÓWKI

Choć w witrynie „Najlepszego sklepu wszech czasów" jako podstawowa rodzina krojów pisma wybrany został font Arial, puryści typograficzni mogą sugerować zastąpienie go krojem *Helvetica* (na jego podstawie zaprojektowany został Arial). Ponieważ większość użytkowników komputerów Macintosh ma zainstalowany font Helvetica, możemy go wykorzystać, aby zaproponować im (a także wszystkim innym posiadaczom tego fontu) ładniejszy wygląd strony. Aby to zrobić, należałoby najpierw wymienić krój Helvetica, a dopiero potem jego zamiennik — Arial, na przykład tak: `body { font-family: Helvetica, Arial, sans-serif; }`.

Różnice pomiędzy tymi dwoma krojami są dla większości osób niedostrzegalne, ale pasjonaci typografii umieją rozróżnić oba kroje z odległości kilometra. Więcej informacji na temat podobieństw obu krojów można uzyskać na stronie projektanta krojów pisma Marka Simonsona: *http://www.ms-studio.com/articlesarialsid.html*.

DODANIE TŁA

Najpierw zadeklarujmy kolory tła dla każdego z pasków. Dzięki temu będziemy widzieli, jakie marginesy i dopełnienia należy zdefiniować.

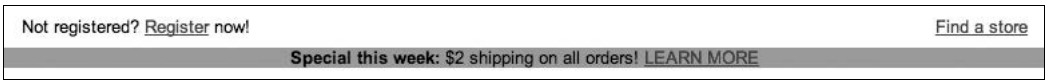
```
#register {
    background: #BDDB62;
}
#message {
    background: #92B91C;
}
```

Rysunek 3.5 ilustruje rezultat zdefiniowania kolorów tła obydwu pasków.

Rysunek 3.5. Dodanie koloru tła ułatwi nam wizualne oszacowanie struktury pasków przed dodaniem pozostałych elementów

POZYCJONOWANIE ZAWARTOŚCI

Następnie zajmiemy się pozycjonowaniem treści. Elementy listy umieścimy na przeciwległych końcach pierwszego poziomego paska, a tekst o promocyjnej cenie wysyłki — pośrodku drugiego paska. Na rysunku 3.6 przedstawiono wygląd komponentu po zastosowaniu zdefiniowanych niżej stylów CSS.



Rysunek 3.6. Elementy listy umieściliśmy na końcach pierwszego paska

```
#register {
    margin: 0;
    padding: 0;
    list-style: none;
    background: #BDDB62;
}
#reg {
    float: left;
    margin: 0;
    padding: 8px 14px;
}
```

```
#find {
    float: right;
    margin: 0;
    padding: 8px 14px;
}

#message {
    clear: both;
    text-align: center;
    background: #92B91C;
}
```

Począwszy od góry: w liście `#register` pozbyliśmy się domyślnych marginesów oraz dopełnienia. Przy użyciu reguły `list-style: none;` wyłączyliśmy także wyświetlanie znaków wypunktowania przy pozycjach listy.

Następnie posłużyliśmy się metodą **przeciwległych obiektów pływających**, aby umieścić elementy listy na dwóch końcach górnego paska. Dla pierwszego elementu (`#reg`) ustawiliśmy wyrównanie do lewej strony, a drugiemu (`#find`) przypisaliśmy wyrównanie do prawej strony. Dzięki takiemu rozwiązaniu możliwe jest umieszczenie dwóch elementów na jednakowej wysokości po przeciwległych stronach paska (rysunek 3.7).

Rysunek 3.7. Metoda przeciwległych obiektów pływających to wygodny sposób wyrównania treści do przeciwległych boków elementu nadrzędnego

Opisana wyżej metoda wyrównania elementów do *przeciwnych* stron to wygodny sposób umieszczenia treści na przeciwległych końcach elementu nadrzędnego.

Wracając do zadeklarowanych stylów — oprócz ustawienia elementów pływających ustawiliśmy także dopełnienie dla każdego boku elementu listy. Dzięki temu zyskaliśmy odpowiednio dużo miejsca, by po lewej stronie odnośnika „Find a store" umieścić ikonę lupy.

Dla drugiego paska dodaliśmy regułę `clear: both;`, dzięki której element ten będzie wyświetlony poniżej elementów pływających. Wyśrodkowanie tekstu akapitu uzyskaliśmy poprzez dopisanie reguły `text-align: center;`.

BRAKUJĄCE TŁO

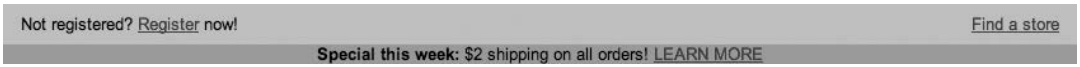
Gdzie podziało się tło górnego paska? Z bardzo podobnym problemem spotkaliśmy się w rozdziale 2. Gdy ustawiamy atrybut `float` dla elementów wewnętrznych (w tym przypadku dwóch elementów `<li>`), wyjmujemy je z normalnego układu dokumentu. Stąd obejmujący pozycje listy element `<ul>` nie wie nawet, czy elementy listy istnieją. W rezultacie `<ul>` nie zna swojej prawidłowej wysokości, a zatem nie wie, na jaką wysokość powinno sięgać jego tło.

Aby naprawić ten problem, listę `<ul>` także ustawiamy jako element pływający (tak, jak zrobiliśmy to w przypadku zakładek z rozdziału 2.). Musimy również zdefiniować szerokość listy (`width`), by mieć pewność, że pasek zajmie cały przeznaczony dla niego obszar. Wygląda na to, że większość przeglądarek dosłownie zaimplementowała wymaganie specyfikacji CSS 2.0: „Element pływający musi mieć jawnie przypisaną szerokość" (*http://www.w3.org/TR/REC-CSS2/visuren.html#floats*). Jeśli nie podamy tutaj szerokości, pasek będzie miał szerokość wymuszoną przez umieszczoną w jego wnętrzu treść (czyli dwie pozycje listy).

```
#register {
    float: left;
    width: 100%;
    margin: 0;
    padding: 0;
    list-style: none;
    background: #BDDB62;
}
#reg {
    float: left;
    margin: 0;
    padding: 8px 14px;
}
#find {
    float: right;
    margin: 0;
    padding: 8px 14px;
}

#message {
    clear: both;
    text-align: center;
    background: #92B91C;
}
```

Na rysunku 3.8 przedstawiono aktualny wygląd pasków — jak widać, tło górnego paska jest ponownie widoczne.



Rysunek 3.8. Gdy przy użyciu właściwości float rozmieścimy elementy znajdujące się wewnątrz kontenera odpowiedzialnego za wyświetlenie dla nich tła, wyświetlanie tła możemy naprawić, ustawiając sam kontener jako element pływający

DODANIE SZCZEGÓŁÓW

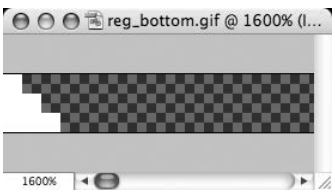
Rysunek 3.9. Zaokrąglony narożnik to po prostu kilka białych pikseli ułożonych w sposób „schodkowy"

Pozostało nam już tylko dodać kilka szczegółów, które pozwolą wykończyć projekt pasków. Zacznijmy od górnego paska i wprowadźmy zaokrąglone narożniki, które widoczne są przy dolnej krawędzi na każdym jego końcu (rysunek 3.9).

Po powiększeniu ilustracji (jak na rysunku 3.9) widać, że *zaokrąglony* róg to tak naprawdę kilka białych pikseli ułożonych na kształt schodów. Gdy obrazek wyświetlany jest w normalnej rozdzielczości, wydaje się, że końce paska są zaokrąglone.

Utworzenie efektu zaokrąglonych narożników to świetna sztuczka, a na dodatek bardzo łatwo można ją zaimplementować z wykorzystaniem małego obrazka i deklaracji koloru tła w arkuszu CSS.

Najpierw w programie Photoshop (lub innym wybranym programie graficznym) utworzymy obrazek tła. Nasze paski mają ustaloną szerokość (768 pikseli), dlatego możemy utworzyć jeden obrazek, w którym umieścimy zarówno lewy, jak i prawy róg paska. Następnie w arkuszu stylów wskażemy ten obrazek jako tło pierwszego paska.



Rysunek 3.10. Lewy koniec obrazka GIF o szerokości 768 pikseli (powiększenie: 1600%)

Rysunek 3.10 przedstawia gotowy obrazek w powiększeniu. Ściślej rzecz biorąc, widzimy na nim tylko lewy koniec obrazka, którego szerokość wynosi 768 pikseli (tej samej szerokości będą obydwa paski). Na obydwu końcach obrazka utworzyliśmy za pomocą narzędzia *Pencil* (*Ołówek*) o grubości 1 piksela biały schodkowy motyw (biały, ponieważ taki jest kolor tła całej strony). Pozostała część obrazka jest przezroczysta (w programie Photoshop przezroczysty obszar zaznaczony jest wzorem szachownicy). Białe kawałki obrazka zostaną umieszczone na kolorowym tle zdefiniowanym w arkuszu stylów. Dzięki temu uzyskamy wrażenie, że końce paska są zaokrąglone.

Do wcześniejszej deklaracji stylu dla elementu o identyfikatorze `#register` (nadrzędnej listy `<ul>`) dodajemy następującą regułę:

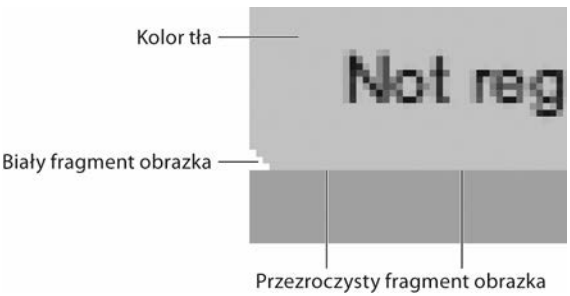
```
#register {
    float: left;
    width: 100%;
```

```
    margin: 0;
    padding: 0;
    list-style: none;
    background: #BDDB62 url(img/reg_bottom.gif) no-repeat
bottom left;
}
```

Zdefiniowaliśmy właśnie kolor tła i umieściliśmy na nim obrazek. Obrazek tła *nie* może być powielany i musi być wyrównany do dolnej i lewej krawędzi elementu. Tło będzie widoczne przez przezroczystą część obrazka, a białe narożniki będą je zasłaniać. Wyrównanie obrazka do dolnej krawędzi sprawia, że bez względu na wysokość elementu (ulegającą zmianie wraz ze zmianą rozmiaru tekstu lub zmianą ilości treści) białe rogi zawsze będą znajdować się w odpowiednim miejscu (rysunek 3.11).

Rysunek 3.11. Trójwymiarowy rysunek obrazujący kolejność nakładania się elementów

Na rysunku 3.12 przedstawiono otrzymany rezultat — dolna krawędź pierwszego paska jest przy końcach zaokrąglona.



Rysunek 3.12. Połączenie koloru tła i białych fragmentów obrazka tła daje efekt zaokrąglonych dolnych narożników paska

WSKAZÓWKI

Standard CSS3 przewiduje możliwość zastosowania kilku obrazków tła dla jednego elementu. Właściwość ta jest obsługiwana w najnowszych przeglądarkach (w tym w IE9). Więcej informacji na ten temat znajdziesz pod adresem *http://www.css3.info/preview/multiple-backgrounds/*.

CZTERY ZAOKRĄGLONE NAROŻNIKI

W przypadku drugiego paska zaokrąglone narożniki powinny być widoczne *zarówno* przy górnej, jak i dolnej krawędzi, a pasek nadal powinno się dać rozciągnąć w pionie. Możemy to osiągnąć, stosując dwa obrazki tła. Przy dolnej krawędzi wykorzystamy obrazek utworzony dla górnego paska, a przy górnej użyjemy tego samego obrazka, ale po odwróceniu go do góry nogami. Aby zastosować dwa obrazki tła, potrzebne są nam dwa osobne elementy, do których będziemy mogli je przypisać.

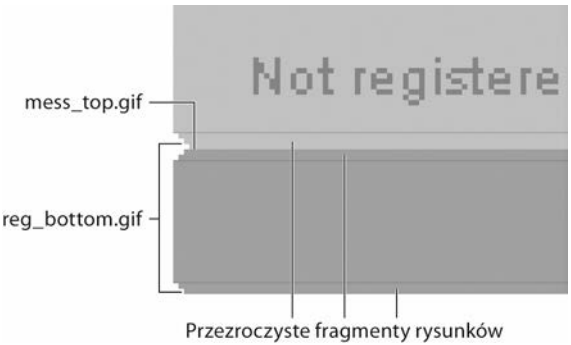
Na szczęście mamy dwa elementy, które możemy wykorzystać. W kodzie odpowiadającym za wyświetlenie drugiego paska mamy element , w którym umieszczony jest akapit <p>:

```
<div id="message">
    <p><strong>Special this week:</strong> $2 shipping
on all orders! <a href="/special/">LEARN MORE</a></p>
</div>
```

Do każdego z tych elementów przypiszemy teraz po jednym obrazku tła. Obrazek z poprzedniej części przykładu przypiszemy do akapitu <p>, a jego odwróconą o 180 stopni wersję przypiszemy do elementu o identyfikatorze #message.

```
#message {
    clear: both;
    text-align: center;
    background: #92B91C url(img/mess_top.gif) no-repeat
top left;
}
#message p {
    margin: 0;
    padding: 8px 14px;
    background: url(img/reg_bottom.gif) no-repeat
bottom left;
}
```

Dzięki wyrównaniu górnych narożników do górnej krawędzi elementu #message (zewnętrzny), a dolnych narożników do dolnej krawędzi elementu <p> (rysunek 3.13) mamy pewność, że wszystkie cztery rogi będą poprawnie rozmieszczone bez względu na to, jak duży lub mały będzie znajdujący się w akapicie tekst. Jeśli powiększymy rozmiar tekstu lub umieścimy w pasku większą ilość tekstu, górne narożniki pozostaną wyrównane do górnego brzegu elementu, a dolne do dolnego brzegu akapitu (jak zobaczymy za chwilę).



Rysunek 3.13. W przypadku dolnego paska zastosowaliśmy dwa obrazki. Kolor tła prześwituje przez przezroczyste obszary rysunków GIF

Na rysunku 3.14. przedstawiono wygląd pasków po dodaniu do arkusza stylów ostatnich deklaracji. Również w przypadku drugiego paska zastosowaliśmy przezroczyste obrazki z białymi narożnikami — przez ich puste miejsca widać tło strony.

Rysunek 3.14. Po dodaniu obrazków tła paski zaczynają nabierać właściwych kształtów

SZCZEGÓŁY ZWIĄZANE Z TEKSTEM I ODNOŚNIKAMI

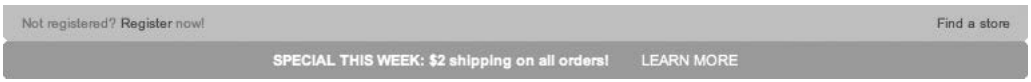
Pozostało nam tylko kilka detali, by całkowicie odwzorować stary projekt — należy jeszcze zmienić kolor tekstu i odnośników. Brakuje także grafik przy odnośnikach „Find a store" oraz „LEARN MORE". Zajmijmy się tym.

Najpierw zdefiniujemy kolor tekstu i odnośników dla każdego paska — w tym celu dodamy potrzebne reguły do istniejących już deklaracji stylów.

```
#register {
    float: left;
    width: 100%;
    margin: 0;
    padding: 0;
    list-style: none;
    color: #690;
    background: #BDDB62 url(img/reg_bottom.gif)
    no-repeat bottom left;
```

```
}
#register a {
    text-decoration: none;
    color: #360;
}
#reg {
    float: left;
    margin: 0;
    padding: 8px 14px;
}
#find {
    float: right;
    margin: 0;
    padding: 8px 14px;
}
#message {
    clear: both;
    font-weight: bold;
    font-size: 110%;
    color: #FFF;
    text-align: center;
    background: #92B91C url(img/mess_top.gif) no-repeat
    top left;
}
#message p {
    margin: 0;
    padding: 8px 14px;
    background: url(img/reg_bottom.gif) no-repeat
bottom left;
}
#message strong {
    text-transform: uppercase;
}
#message a {
    margin: 0 0 0 6px;
    padding: 2px 15px;
    text-decoration: none;
    font-weight: normal;
    color: #FFF;
}
```

Ustawiliśmy kolor tekstu dla odnośników umieszczonych w elemencie `#register` oraz domyślną wielkość czcionki i kolor tekstu dla odnośników i tekstu znajdujących się wewnątrz elementu `#message` (rysunek 3.15).



Rysunek 3.15. Wygląd pasków po dodaniu stylów dla odnośników i tekstu. Widoczne jest także wolne miejsce pozostawione na ikony, które będą towarzyszyć odnośnikom „Find a store" oraz „LEARN MORE"

Tekst „Special this week:" został przez nas wcześniej wyróżniony przy użyciu elementu `<strong>`. Teraz skorzystaliśmy z możliwości właściwości `text-transform` i przekształciliśmy wyróżniony tekst na wersaliki, utrzymując oryginalny tekst w kodzie, napisany zgodnie z *normalną wielkością* liter w zdaniu. Dlaczego jest to dobre rozwiązanie? Otóż o ile teraz chcemy, by napisy „SPECIAL THIS WEEK" oraz „LEARN MORE" były złożone wielkimi literami, o tyle kiedyś może się to zmienić. Może się zdarzyć, że nowy dyrektor artystyczny projektu zażyczy sobie, by zmienić styl tego tekstu na *same małe litery*. Dzięki właściwości `text-transform` możemy z łatwością wprowadzić takie poprawki, uaktualniając *sam* arkusz stylów CSS i nie dotykając kodu źródłowego strony.

To kolejny przykład przygotowania się na alternatywne wersje, jakie może w przyszłości przybrać projekt strony. **Należy pamiętać, by tekst z kodu źródłowego był rozdzielony od warstwy prezentacyjnej,** i wykorzystywać właściwość `text-transform` do zmiany tekstu na małe lub wielkie litery w zależności od aktualnych potrzeb.

OSTATNI ETAP

Ostatni etap przebudowy poziomych pasków będzie polegał na dodaniu grafik wyświetlanych obok odnośników „Find a store" oraz „LEARN MORE". Obrazki te mogliśmy dodać w kodzie strony, jednak w celu ułatwienia późniejszej aktualizacji wszelkie mało istotne elementy graficzne lepiej jest trzymać poza kodem źródłowym dokumentu — zwłaszcza że możemy z łatwością użyć ich jako obrazków tła z wykorzystaniem stylów CSS.

Najpierw do drugiego elementu listy w górnym pasku dodajmy ikonę przedstawiającą lupę. Pozycję grafiki definiujemy wartościami 0 50%, dzięki czemu zostanie ona umieszczona przy lewym brzegu elementu, w połowie wysokości, licząc od górnej krawędzi (czyli zostanie wyśrodkowana w pionie):

```
#find {
    float: right;
    margin: 0;
```

```
    padding: 8px 14px;
    background: url(img/mag-glass.gif) no-repeat 0 50%;
}
```

Na rysunku 3.16 widoczny jest efekt dodania ikony do odnośnika „Find a store".

Rysunek 3.16. Obrazek tła został wyrównany do lewej krawędzi elementu listy, gdzie wcześniej odpowiednio zwiększyliśmy dopełnienie

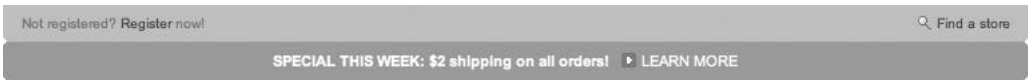
Teraz obok odnośnika „LEARN MORE" znajdującego się w drugim pasku dodamy grafikę strzałki. Pozycjonujemy ją jak wcześniej — przy użyciu wartości 0 50%, czyli wyrównamy ją do lewej strony i do połowy wysokości elementu:

```
#message a {
    margin: 0 0 0 6px;
    padding: 2px 15px;
    text-decoration: none;
    font-weight: normal;
    color: #fff;
    background: url(img/arrow.gif) no-repeat 0 50%;
}
```

Na rysunku 3.17 pokazano wygląd elementu po dodaniu ikony strzałki po lewej stronie odnośnika „LEARN MORE". Strzałka została umieszczona na stronie poprzez zadeklarowanie stylu dla elementu <a> znajdującego się wewnątrz drugiego paska.

Na rysunku 3.18 przedstawiono ostateczny wygląd przykładu — wszystkie elementy znajdują się we właściwym miejscu. Otrzymaliśmy dwa paski praktycznie identyczne z tymi, które widoczne były w witrynie „Najlepszego sklepu wszech czasów", jednak kryjąca się za naszym projektem struktura kodu i wszystkie zabiegi, których dokonaliśmy, by strategicznie rozmieścić obrazki tła i tekst, sprawiają, że projekt jest *kuloodporny*. A właściwie prawie kuloodporny — potrzebujemy jeszcze jednej poprawki, by wszystko działało w przeglądarce Internet Explorer 7.

Rysunek 3.17. W ten sam sposób, w jaki dodaliśmy obrazek do elementu listy w pierwszym pasku, po lewej stronie tekstu LEARN MORE dodaliśmy rysunek strzałki. Tym razem obrazek został dodany bezpośrednio do elementu <a>



Rysunek 3.18. Ostateczna, kuloodporna wersja projektu poziomych pasków

POPRAWKA DLA IE7

Jeśli gotowy projekt wyświetlimy w przeglądarce Internet Explorer 7, zauważymy dodatkowy odstęp ponad tekstem w dolnym pasku (rysunek 3.19). Grrr! Z nieznanych powodów IE7 dodaje ten odstęp, choć inne przeglądarki (w tym jej poprzednik, IE6) tego nie robią. Najłatwiejszą (i najmniej uciążliwą) metodą poradzenia sobie z tym problemem jest skorzystanie z dodatkowego elementu pływającego, zgodnie z techniką opisaną wcześniej. Element o identyfikatorze #reg już wcześniej został zadeklarowany jako pływający, aby zmieściły się w nim dwa pływające elementy listy; teraz zastosujemy jeszcze regułę `float: left;` do elementu o identyfikatorze #message, by poradzić sobie z dodatkowym odstępem w IE7.

Rysunek 3.19. Gotowy przykład wyświetlony w przeglądarce Internet Explorer 7, pokazujący dodatkowy odstęp ponad treścią drugiego paska

Poprawiona deklaracja będzie wyglądała następująco:

```
#message {
    float: left;
    width: 100%;
```

```
    margin: 0;
    padding: 0;
    font-weight: bold;
    font-size: 110%;
    color: #fff;
    text-align: center;
    background: #92B91C url(img/mess_top.gif) no-repeat
    top left;
}
```

Tak jak wcześniej, dodaliśmy również regułę `width: 100%`, niezbędną do rozciągnięcia paska na całą szerokość kontenera (w tym przypadku 768 pikseli). Takie wyjście jest proste, szybkie i nadaje się do zastosowania we wszystkich przeglądarkach. W tym przykładzie jest to chyba najlepsze rozwiązanie problemu występującego w IE7. Istnieją jednak również inne sposoby radzenia sobie z problemami z elementami pływającymi. W kolejnym rozdziale omówimy inną metodę, która pozwala utrzymywać niezależność elementów, tak by nie reagowały one na to, co dzieje się *po nich* w strukturze dokumentu.

Dlaczego rozwiązanie to jest kuloodporne

Po uproszczeniu kodu i wykorzystaniu w kreatywny sposób małych obrazków tła z powodzeniem odtworzyliśmy oryginalny projekt. Teraz dowiemy się, *dlaczego* zaproponowane rozwiązanie jest takie dobre, czyli elastyczne.

ODDZIELENIE STRUKTURY OD WYGLĄDU

Wyrzuciliśmy zbędne grafiki i tabele, a na ich miejsce wprowadziliśmy schludny, ustrukturyzowany kod HTML. Sensownie zastosowane znaczniki będą lepiej zinterpretowane przez więcej urządzeń i programów, nawet w przypadku ewentualnego braku obsługi stylów CSS.

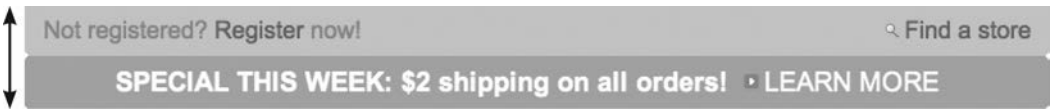
Obrazki wykorzystane w projekcie umieściliśmy w arkuszu stylów, zamiast wstawiać je bezpośrednio do kodu. Dzięki temu wprowadzenie późniejszych zmian w szacie graficznej będzie znacznie prostsze, nie wspominając już o tym, że poprzez zastosowanie takiego rozwiązania znacznie ograniczyliśmy *ilość* kodu.

Zmiana koloru pasków, na przykład na czerwony, niebieski czy jakikolwiek inny, będzie polegała na prostej modyfikacji kilku reguł CSS. A rezultat będzie widoczny *natychmiast*.

KONIEC Z USTALONYMI WYSOKOŚCIAMI

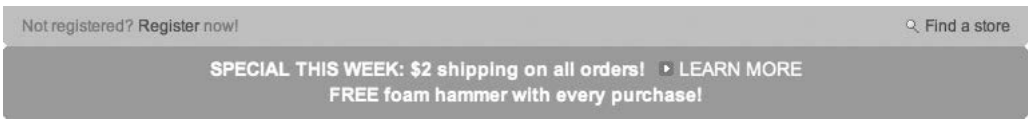
Zamiast zakładać, że paski zawsze będą miały wysokość x pikseli, w pomysłowy sposób rozmieściliśmy obrazki tła, dzięki czemu zachowaliśmy integralność projektu, a jednocześnie umożliwiliśmy dowolne rozciąganie i zwężanie paska. Takie rozwiązanie pozwala także na stosowanie dowolnych metod definiowania rozmiaru tekstu (takich jak te opisane w rozdziale 1., „Elastyczny tekst") — nie trzeba już uważać, by wymiary tekstu podane w pikselach nie przekroczyły rozmiaru paska.

Na rysunku 3.20 przedstawiono zrekonstruowany projekt wyświetlony przy znacznie powiększonym rozmiarze tekstu. Tło i zaokrąglone narożniki pasków nadal wyglądają dobrze i znajdują się we właściwym miejscu.



Rysunek 3.20. Po zwiększeniu rozmiaru tekstu paski odpowiednio się rozciągają, a projekt od strony graficznej nadal wygląda bez zarzutu

Wyobraźmy sobie teraz, że redaktor chce w pasku promocyjnym umieścić dwie wiadomości. Bez modyfikowania projektu możemy w prosty sposób dodać drugi wiersz tekstu — pasek sam dopasuje się do nowego wymiaru wiadomości (rysunek 3.21). Przykład ten ilustruje największą zaletę stosowania rozwiązań kuloodpornych — projekt dopasowuje się nawet do *nieprzewidzianych* sytuacji.



Rysunek 3.21. Dodanie drugiego wiersza tekstu do elementu #message nie wymaga żadnych przeróbek ze strony projektanta, ponieważ strona została opracowana pod kątem możliwości wyświetlenia dowolnej ilości treści

Jeśli nasz klient lub szef powie: „W tym miejscu będziemy umieszczać najwyżej jeden wiersz tekstu", a my wykonamy projekt zgodnie z taką specyfikacją, to możemy być prawie pewni, że za tydzień ta sama osoba stwierdzi, iż „oczywiste jest, że tutaj niezbędne jest miejsce na *dwa* wiersze tekstu". Jeśli tworzymy kuloodporne projekty, taką funkcjonalność będziemy mieli od razu wbudowaną w projekt. Oczywiście szefowi czy klientowi należy powiedzieć, że przygotowanie miejsca na drugi wiersz tekstu zajęło nam kolejny tydzień. Albo pochwalić się

WSKAZÓWKI

Rysunek lupy i strzałki umieściliśmy w połowie wysokości elementu nadrzędnego. Warto zauważyć, że dzięki takiemu rozwiązaniu obrazki — bez względu na rozmiar tekstu — zawsze będą wyśrodkowane w pionie względem tekstu.

swoją przezornością i powiedzieć, że od razu pomyśleliśmy o odpowiednim rozwiązaniu. Przy okazji możemy pokazać, w jaki sposób stosowanie CSS ogranicza problemy związane z aktualizacją projektu.

Wariant z atrybutem border-radius

Ponieważ oryginalny projekt, który odtwarzaliśmy, został zaprojektowany z użyciem bitmapowych narożników „w schodki", my także użyliśmy malutkich elementów graficznych, które wiernie naśladowały zaokrąglenie narożników na obydwu paskach. Możemy jednak pokusić się o uproszczenie tego projektu poprzez całkowite wyeliminowanie elementów graficznych i zastąpienie ich atrybutem `border-radius`, dostępnym w CSS3. Przeglądarki, które nie obsługują atrybutu `border-radius`, wyświetlą paski w postaci prostokątów, ale w wielu przypadkach jest to akceptowalny kompromis. Czy zaokrąglone narożniki są *kluczowe* dla danego projektu? Jeśli odpowiedź brzmi „nie", to rezygnacja z elementów bitmapowych na rzecz stylów CSS może być znakomitym rozwiązaniem. Większa elastyczność i łatwość wprowadzenia modyfikacji mniejszym nakładem pracy. Czegoż chcieć więcej?

Zapewne pamiętasz, że w rozdziale 2. użyliśmy gradientów CSS3 w celu odtworzenia gradientów na zakładkach nawigacji. Podobnie jak gradienty, atrybut `border-radius` jest obsługiwany przez znakomitą większość nowoczesnych przeglądarek (Safari, Firefox, Opera, IE9+). Trzeba jedynie pamiętać, aby zastosować odpowiednie prefiksy atrybutów w celu uzyskania spójnego wyglądu zaokrąglonych narożników w różnych przeglądarkach.

Zastosujmy atrybut `border-radius` na przykładzie omówionym w tym rozdziale, aby wypróbować go w praktyce. Przy okazji pozbędziemy się przygotowanych uprzednio narożników. Wystarczy, że zmodyfikujemy dwie deklaracje stylów, by zaokrąglić lewy oraz prawy dolny narożnik w elemencie `#register` oraz wszystkie cztery narożniki w elemencie `#message`. Usuniemy też odwołania do obrazków z reguł związanych z atrybutem `background`.

```
#register {
  float: left;
  width: 100%;
  margin: 0;
  padding: 0;
  list-style: none;
```

```
    color: #690;
    background: #BDDB62;
    -webkit-border-bottom-left-radius: 5px;
    -webkit-border-bottom-right-radius: 5px;
    -moz-border-radius-bottomleft: 5px;
    -moz-border-radius-bottomright: 5px;
    border-bottom-left-radius: 5px;
    border-bottom-right-radius: 5px;
    }
  #message {
    float: left;
    width: 100%;
    margin: 0;
    padding: 0;
    font-weight: bold;
    font-size: 110%;
    color: #FFF;
    text-align: center;
    background: #92B91C;
    -webkit-border-radius: 5px;
    -moz-border-radius: 5px;
    border-radius: 5px;
    }
```

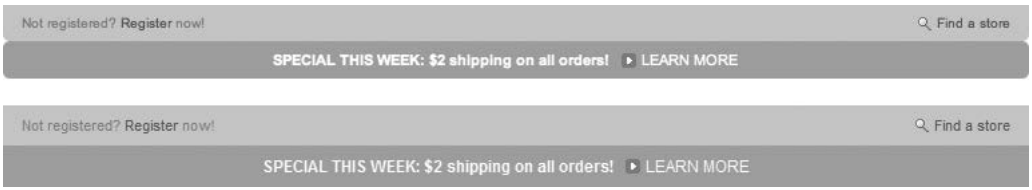
Zastosowaliśmy taką składnię dla atrybutu `border-radius`, która powinna bez zarzutu zadziałać w przeglądarkach opartych na silniku WebKit (Safari, Chrome) i Mozilla (Firefox). Dodatkowo reguły pozbawione prefiksów gwarantują uzyskanie poprawnego efektu w Operze oraz IE9+ (a także wszystkich przyszłych przeglądarkach, które będą obsługiwały atrybut `border-radius`).

Warto zauważyć, że składnia zaokrąglania lewego i prawego dolnego narożnika różni się nieznacznie w implementacjach `–webkit-` oraz `–moz-`. W razie problemów z zapamiętaniem tych różnic dodaj stronę *http://border-radius.com* do ulubionych — znajdziesz na niej skrypt, który wygeneruje odpowiedni kod za Ciebie.

Rysunek 3.22 (na następnej stronie) przedstawia porównanie pięknie zaokrąglonych narożników w przeglądarce Safari (rysunek u góry) z kwadratowymi, niezaokrąglonymi narożnikami w IE8, która nie obsługuje atrybutu `border-radius` (rysunek na dole). Czy brak zaokrąglonych narożników w niektórych przeglądarkach oznacza koniec świata? Raczej nie, zwłaszcza że elastyczność wynikająca z rezygnacji z obrazków wyłącznie w celach estetycznych często przeważa nad ewentualnymi uciążliwościami związanymi z brakiem obsługi opisywanego atrybutu w starszych przeglądarkach.

WSKAZÓWKI

Reguły bez prefiksów odwołujących się do silników przeglądarek zawsze powinny być podawane na początku. Dzięki takiemu rozwiązaniu strona zostanie poprawnie wyświetlona nawet w przypadku ewentualnych zmian w specyfikacji, dotyczących (niekiedy) eksperymentalnych atrybutów.



Rysunek 3.22. Zaokrąglone narożniki w przeglądarce Safari (u góry) oraz niezaokrąglone w IE8 (na dole)

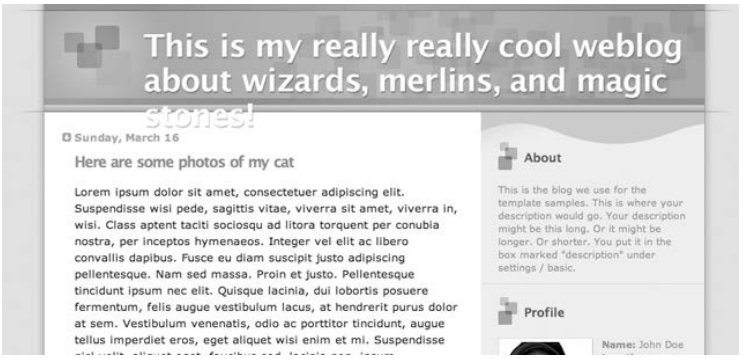
Inny przykład rozciągania

Inne zastosowanie elastycznych elementów projektu przedstawię na przykładzie procesu tworzenia nagłówka dla szablonu TicTac, który zaprojektowałem dla witryny Blogger. Blogger to popularne narzędzie do publikowania blogów, należące do firmy Google. Oferuje ono kilka gotowych szablonów, które użytkownicy mogą wykorzystać przy tworzeniu swojego dziennika internetowego. Projekty szablonów musiały być kuloodporne, aby nagłówek mógł pomieścić dowolnie długi tytuł witryny.

Na rysunku 3.23 widoczny jest nagłówek szablonu i wyświetlony w nim tytuł „Sample Blog". Wszystkie elementy graficzne nagłówka to obrazki tła zdefiniowane w arkuszu stylów CSS. Tytuł witryny musi być *zwykłym tekstem*, tak by użytkownicy serwisu Blogger mogli skorzystać z szablonów bez konieczności tworzenia jakiejkolwiek własnej grafiki.

Rysunek 3.23. Przykładowa strona w witrynie Blogger utworzona z wykorzystaniem szablonu TicTac. W tym szablonie elastyczność jest wręcz wymogiem, ponieważ tekst nagłówka może mieć dowolną długość

W przypadku jednowierszowych tytułów przy określonej wielkości czcionki projekt nagłówka wygląda świetnie. Jednak co się stanie, jeśli tytuł bloga będzie dłuższy (rysunek 3.24)? Gdybym zdefiniował stałą wysokość dla nagłówka, długie tytuły (lub złożone tekstem o większym rozmiarze) wystawałyby poza jego tło i stałyby się nieczytelne, co popsułoby cały projekt.



Rysunek 3.24. Uzyskany rezultat może być bardzo nieciekawy, gdy stałą wysokość elementów połączymy z tekstami o różnej długości

Integralną częścią każdego dobrego projektu szablonu jest możliwość pomieszczenia treści dowolnej długości. Aby nagłówek w szablonie dla witryny Blogger dawał się rozciągać, posłużyłem się stylami CSS i w sprytny sposób rozmieściłem dwa obrazki tła.

STRUKTURA KODU

Zanim w ogóle pomyślałem o stylach i grafice, ustaliłem strukturę kodu dla nagłówka. Musiałem mieć możliwość użycia dwóch różnych obrazków tła, dlatego potrzebowałem w kodzie dwóch elementów, do których mógłbym przypisać te obrazki:

```
<header role="banner">
    <h1>Sample Blog</h1>
</header>
```

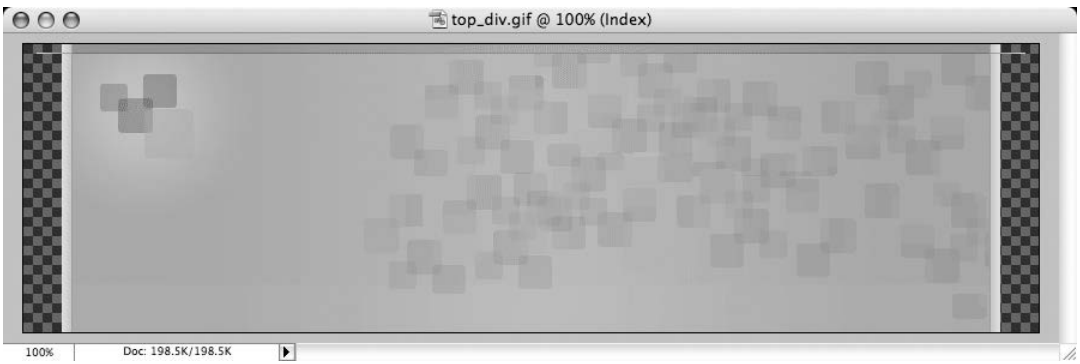
Tytuł bloga postanowiłem zdefiniować jako element <h1>. Jeśli w naszym projekcie chcemy oszczędzić ten element dla innych zastosowań, oczywiście możemy w tym miejscu użyć dowolnego innego. Ważne jest, byśmy mieli do dyspozycji dwa osobne elementy. Dlatego element <h1> umieściłem w dodatkowym elemencie HTML5 o nazwie <header>. Ponownie skorzystałem przy tym z możliwości definiowania ról WAI-ARIA. Deklaracja `role="banner"` oznacza, że ten konkretny

element <header> (konkretny, gdyż może on zostać wykorzystany na stronie więcej niż raz) jest *najważniejszym* nagłówkiem strony. W tym przypadku jest to idealne rozwiązanie. Atrybut `role` nie tylko poprawia dostępność projektu, lecz stanowi też dodatkowy uchwyt dla stylów, który można wykorzystać zamiast identyfikatora `id`.

TWORZENIE DWÓCH OBRAZKÓW

Aby uzyskać możliwość elastycznego rozciągania nagłówka, utworzyłem dwa obrazki. Jeden z nich był o wiele wyższy, niż *uważałem*, że jest to konieczne. Użytkownicy serwisu Blogger odkrywają większą lub mniejszą jego część w zależności od długości tytułu bloga. Drugi obrazek to dolne obramowanie nagłówka. Obrazek ten zawsze będzie wyświetlany *pod tekstem* nagłówka.

Na rysunku 3.25 widoczny jest większy z obrazków — przez prawie całą jego wysokość powtarzany jest jeden motyw.

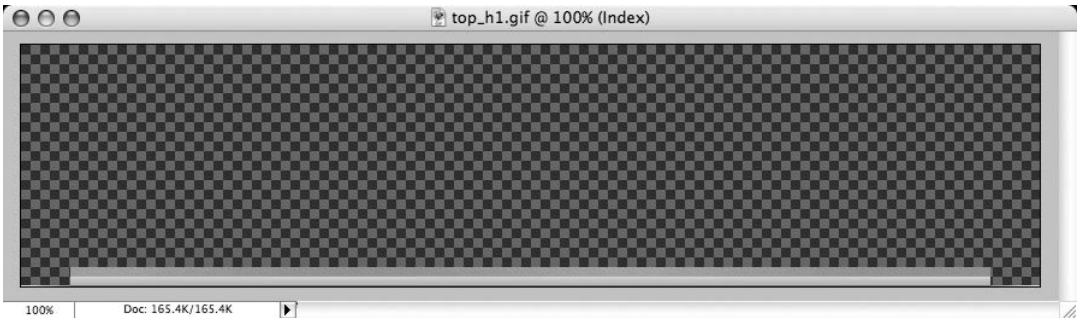


Rysunek 3.25. Pierwszy obrazek ma dużo większą wysokość, niż wydaje się potrzebna

Na rysunku 3.26 przedstawiono drugi obrazek — podkreślenie, które zawsze będzie umieszczone *poniżej* tekstu tytułu. Cała górna część obrazka jest przezroczysta. Dzięki temu możliwe jest nałożenie na siebie dwóch obrazków — obrazek *top_div.gif* będzie widoczny spod obrazka *top_h1.gif*.

ZASTOSOWANIE STYLÓW CSS

Do połączenia wszystkich elementów w jedną całość posłużyłem się dwiema raczej prostymi deklaracjami CSS. Najpierw zadeklarowałem style dla zewnętrznego elementu <header>:



Rysunek 3.26. Obrazek top_h1.gif stanowi graficzne podkreślenie nagłówka. Pozostała jego część jest przezroczysta, dzięki czemu umieszczone pod nim obrazki także będą widoczne

```
header[role="banner"] {
    margin: 0;
    padding: 0;
    font-family: "Lucida Grande", "Trebuchet MS";
    background: #E0E0E0 url(img/top_div.gif) no-repeat
top left;
}
```

Jak widać, zdefiniowałem domyślną rodzinę fontów, wyrównałem obrazek *top_div.gif* do górnej oraz lewej krawędzi i zastosowałem jasnoszary kolor jako tło.

Następnie dodałem deklaracje dla elementu nagłówka:

```
header[role="banner"] {
    margin: 0;
    padding: 0;
    font-family: "Lucida Grande", "Trebuchet MS";
    background: #E0E0E0 url(img/top_div.gif) no-repeat
    top left;
}
header[role="banner"] h1 {
    margin: 0;
    padding: 45px 60px 50px 160px;
    font-size: 200%;
    color: #fff;
    background: url(img/top_h1.gif) no-repeat bottom
    left;
}
```

Powyższe deklaracje CSS definiują wartości atrybutów `font-size` oraz `padding` dla tekstu i wyrównują obrazek *top_h1.gif* do *dolnej* oraz lewej krawędzi nagłówka. Wielkość elementu <header> zależy

wyłącznie od wysokości umieszczonego w nim tekstu, więc za tekstem nagłówka wyświetlony będzie tylko potrzebny fragment obrazka tła.

Na rysunku 3.27 zilustrowano, w jaki sposób poszczególne elementy się na siebie nakładają.

Rysunek 3.27. Rozmieszczenie elementów projektu — górna część rozciąga się tak, by pomieścić cały tytuł

AUTOROZCIĄGANIE

Spróbujmy teraz przetestować nasz rozciągliwy nagłówek — umieśćmy w nim bardzo długi tytuł. Jak widać na rysunku 3.28, przy długim tytule odsłonięta została większa część obrazka *top_div.gif*, a obrazek *top_h1.gif* został przesunięty w dół, tak by nadal znajdował się pod tekstem tytułu.

Rysunek 3.28. Klucz do sukcesu polega na przewidzeniu możliwości umieszczenia w projekcie dowolnej ilości tekstu

Utworzony w taki sposób nagłówek może się także skurczyć. Jeśli tekst tytułu jest krótki lub właściciel strony chce, by był on wyświetlony przy użyciu małej czcionki, pole nagłówka *zmniejszy się* do odpowiednich wymiarów. Krótko mówiąc, szablon jest wyposażony w estetyczny wizualnie nagłówek, który został przygotowany na każdą ewentualność.

Podsumowanie

W języku angielskim funkcjonuje ciekawe określenie, a mianowicie *piekarski tuzin*, oznaczające ni mniej, ni więcej… trzynaście. Chodzi w nim o to, że mądry piekarz zawsze upiecze dodatkowe ciastko czy bułkę. Jeśli jedno ciastko się przypali albo nie wyrośnie (lub zostanie zjedzone przez piekarza), zawsze będzie jedno dodatkowe, gdyby potrzebny był *cały tuzin*. W ten sposób piekarz przygotowuje się na niespodziewane okoliczności. Dlaczego my nie mielibyśmy upiec dodatkowego ciastka, gdy i tak pieczemy całą blachę? Czasami jedno dodatkowe ciastko jest bardzo potrzebne, czasami nie.

Odłóżmy ciastka na bok (ale oczywiście możemy pozostawić je w zasięgu ręki). Uwzględnienie w naszym projekcie możliwości powiększenia się w pionie poziomych elementów strony możemy porównać z przygotowaniem piekarskiego tuzina ciastek. W ten sposób próbujemy zabezpieczyć się przed niespodziewanymi sytuacjami: przygotowujemy dodatkowe miejsce na powiększenie tekstu lub dodanie kilku wyrazów. Poprzez to oszczędzamy własny czas, dajemy użytkownikowi (a także redaktorom strony) większą kontrolę nad sposobem wyświetlania strony i poprawiamy jej dostępność.

Oto lista kilku wskazówek, o których warto pamiętać podczas tworzenia poziomych elementów projektu:

- Usunięcie z kodu mało istotnych elementów graficznych i zastąpienie ich obrazkami tła definiowanymi w arkuszu stylów pozwala uniknąć niepotrzebnego rozrastania się kodu.
- Do rozmieszczenia elementów po przeciwnych bokach pojemnika należy stosować metodę „przeciwległych elementów pływających".
- Gdy nie wiemy, jak duża ilość treści może być umieszczona w jakimś elemencie, zawsze możemy po-służyć się dwoma obrazkami tła, by element ten mógł się rozciągać i kurczyć.
- Jeśli pewne zabiegi stylistyczne, takie jak zaokrąglone narożniki, nie są kluczowe dla projektu, możemy skorzystać z atrybutów w rodzaju `border-radius`.
- Planujmy miejsce *z zapasem*. Postępujmy tak, jak przy pieczeniu czekoladowych ciastek: zawsze warto zrobić jedno dodatkowe.

Skorowidz

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

T

U

W

Z

Sięgnij po tę książkę i twórz niezawodne strony w sieci!

Każdego dnia w sieci pojawiają się tysiące nowych witryn. Niejednokrotnie są one perfekcyjne merytorycznie i interesujące wizualnie. Niestety, wiele z nich nie odniesie sukcesu, bo nie trafi do szerokiego grona odbiorców. Konkurencja jest dziś tak ogromna, że Twoje rozwiązanie musi się wyróżniać nie pod jednym, a pod kilkoma względami. Jedną z najważniejszych zalet dobrej strony jest lekki, poprawny kod, korzystający z nowości języka HTML5 i CSS3 oraz dostępny dla różnych urządzeń (stacjonarnych i mobilnych). W trakcie lektury kolejnej edycji tej wyjątkowej książki dowiesz się, jak zagwarantować czytelność i atrakcyjność Twojej strony, nawet jeśli nie masz dostępu do elementów graficznych i CSS. Ponadto zobaczysz, jak sobie radzić z rozmiarem czcionek czy ograniczoną przestrzenią.

Każdy rozdział rozpoczyna się opisem rozwiązania, które nie jest kuloodporne. Rozwiązanie takie przeważnie opiera się na tradycyjnych technikach i zwykłym kodzie HTML. Dan Cederholm, autor tej bestsellerowej pozycji, rozkłada je na czynniki pierwsze i pokazuje przy okazji różne ograniczenia. Następnie proponuje zastosowanie wariantu alternatywnego, opracowanego przy użyciu HTML-a oraz CSS. Ten wariant umożliwia zastąpienie opasłego źródła odchudzonym, strukturalnym kodem oraz przemyślnie opracowanymi regułami CSS. W rezultacie otrzymany projekt jest lekki i dostępny dla wielu użytkowników. Na koniec Dan prezentuje proces tworzenia kompletnej strony internetowej z komponentów omówionych w poprzednich rozdziałach. Ten sposób naprawdę pozwala się wiele nauczyć!

- Skalowanie tekstu za pomocą słów kluczowych, wartości procentowych oraz jednostek em, dające użytkownikowi pełną kontrolę nad czytelnością projektu
- Uwzględnianie elastycznego wydłużania poziomych komponentów strony w pionie
- Zastosowanie elementów pływających do opracowania struktur tabelarycznych
- Zapewnienie czytelności strony nawet wtedy, gdy brak obrazków i obsługi CSS
- Rozdzielanie warstwy prezentacyjnej i kodu w tradycyjnych tabelach oraz odtwarzanie ich wyglądu za pomocą CSS
- Zastosowanie filozofii stopniowego ulepszania projektów dzięki HTML5 oraz CSS3

Nr katalogowy: 10418

Księgarnia internetowa:
http://helion.pl

Zamówienia telefoniczne:
0 801 339900
0 601 339900

helion.pl
księgarnia internetowa

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Cena: 49,00 zł
ISBN 978-83-246-5120-7
9 788324 651207

Informatyka w najlepszym wydaniu