CSS. Projektowanie profesjonalnych stron WWW

Autor: Ch. Schmitt, T. Dominey, C. Li, E. Marcotte, D. Orchard, M. Trammell
Tłumaczenie: Łukasz Piwko
ISBN: 978-83-246-2067-8
Tytuł oryginału: Professional CSS: Cascading Style Sheets for Web Design
Format: 168x237, stron: 320

Twórz profesjonalne strony WWW, korzystające z CSS!

- Dlaczego należy oddzielać warstwę prezentacji od struktury?
- Jak stworzyć różne układy strony z wykorzystaniem CSS?
- Jak wykorzystać format graficzny PNG?

W 1996 roku projektanci stron WWW mogli odetchnąć z ulgą. To właśnie wtedy pojawiła się pierwsza oficjalna dokumentacja CSS, a z nią nadzieja na lepsze, łatwiejsze i bardziej uniwersalne zarządzanie wyglądem stron. Trzeba pamiętać, że wcześniej każdy z producentów przeglądarek wprowadzał swoje własne znaczniki, pozwalające kontrolować kolory czy typografię. Oczywiście producenci robili to bez porozumienia z innymi – łatwo więc sobie wyobrazić, jaki stanowiło to kłopot dla projektantów stron WWW. Na szczęście nadszedł wspomniany rok 1996, a w latach następnych pojawiły się nowsze przeglądarki, coraz lepiej wspierające kaskadowe arkusze stylów.

Dzięki tej książce poznasz najlepsze praktyki związane z CSS oraz XHTML. Na przykładzie prawdziwych witryn dowiesz się, jak zapewnić różne funkcjonalności za pomocą kaskadowych arkuszy stylów. Analityczny opis strony Blogger.com pozwoli zaprezentować różne efekty, choćby takie, jak „rollover" dla tekstu, łączy oraz tabel. Z analiz dotyczących innych witryn dowiesz się, jak zapewnić efekt cienia, rozwijane menu czy też dynamiczne przełączanie arkuszy CSS. Ponadto poznasz sposoby wykorzystania przezroczystych grafik w formacie PNG, zasady tworzenia układów CSS oraz metody pozycjonowania elementów na stronie. Znajdziesz tu także wiele innych zagadnień, które pozwolą Ci zbudować własną – profesjonalną i atrakcyjną – witrynę WWW.

- Deklarowanie typów dokumentu
- Oddzielenie warstwy prezentacji od struktury
- Typy selektorów
- Wykorzystanie efektu „rollover"
- Tworzenie cieni oraz rozwijanego menu
- Zapewnienie poprawności semantycznej
- Techniki przełączania CSS
- Wykorzystanie formatu PNG w różnych przeglądarkach
- Przygotowywanie układów przy użyciu CSS
- Sposoby rozwiązywania problemów z kaskadowymi arkuszami stylów
- Rozwiązywanie problemów z przeglądarkami

Stwórz własną, profesjonalną stronę WWW!

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

3

Witryna internetowa mistrzostw PGA

Wiem, że coraz lepiej gram w golfa, ponieważ rzadziej trafiam w widzów.
— Gerald Ford

Mistrzostwa PGA są jednym z najważniejszych wydarzeń sportowych. Odbywające się co roku pod koniec lata mistrzostwa PGA Ameryki są ostatnim ważnym wydarzeniem sezonu golfowego, przybywają na nie najlepsi profesjonalni golfiści oraz miliony fanów z całego świata.

Turner Sports Interactive, oddział firmy Time Warner, był odpowiedzialny za rozwój witryny i jej treść podczas mistrzostw. Celem było utworzenie dynamicznej, bogatej i zgodnej ze standardami witryny przy użyciu CSS, łatwo przyswajalnego kodu XHTML oraz dodatków specjalnych w technologii Flash. Z kreatywnego punktu widzenia witryna miała mieć unikalny wygląd, powinna być wyrafinowana, a przede wszystkim należało ustrzec się powielania oklepanych motywów z typowych witryn poświęconych golfowi. Zdecydowano się na paletę słabo nasyconych i neutralnych barw oraz kolory czarny i biały w celu podkreślenia oryginalnych fotografii i skierowania uwagi na zawartość tekstową.

Niedługo po zaprezentowaniu w lipcu 2004 roku witryna mistrzostw PGA uzyskała świetne opinie fanów golfa z całego świata i społeczności projektantów sieciowych za unikalny wygląd, trzymanie się standardów oraz ogólny projekt. Liczba odwiedzających wzrosła niebotycznie już pierwszego dnia zawodów, gdy redaktorzy PGA.com zaczęli dostarczać wiadomości, informacje o punktacji oraz treść multimedialną w godzinnych odstępach.

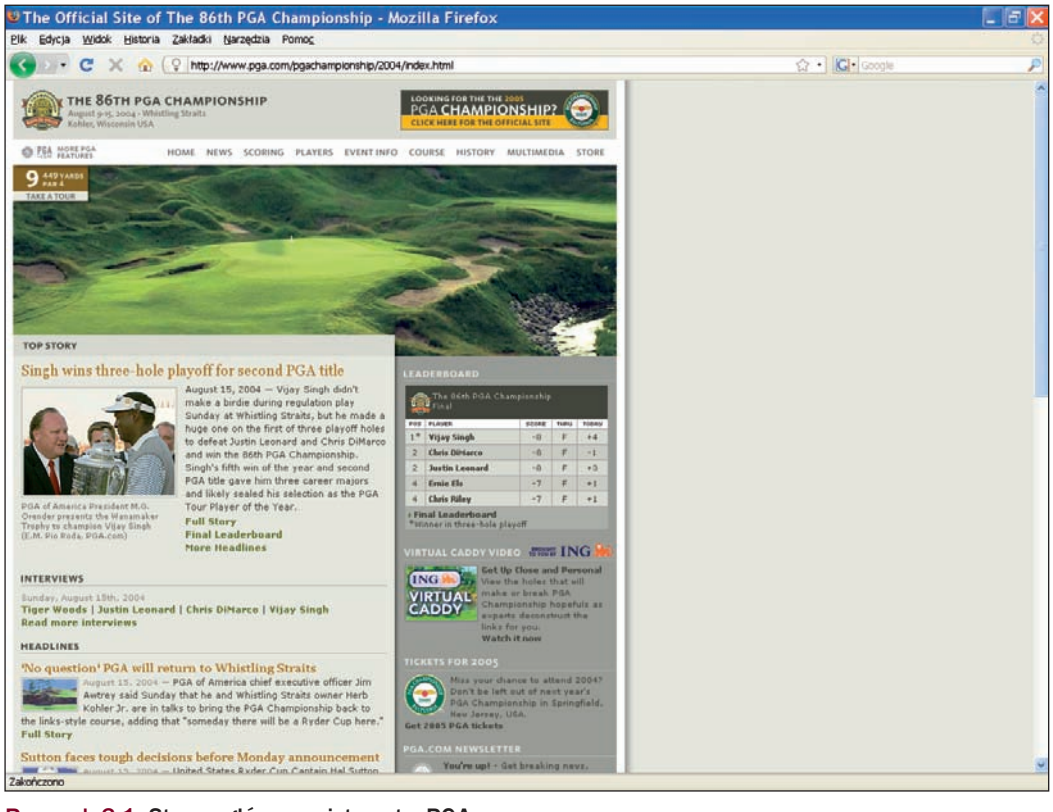
Od pojawienia się tej wersji witryny mistrzostw PGA minęło już kilka lat, przewinęło się też kilka różnych wersji tego projektu. Skupimy się jednak na roku 2004. Utworzona wtedy witryna była klasycznym, solidnym projektem z niesamowitymi fotografiami pól golfowych. Osiągnięto to — oczywiście — przy użyciu CSS i XHTML.

W rozdziale szczegółowo opiszę niektóre z technik zastosowanych w kilku miejscach oryginalnej wersji tej witryny. Nie zabraknie ogólnych wskazówek i sztuczek CSS i XHTML oraz opisu niektórych pułapek, których należy wystrzegać się we własnych projektach. Oto lista poruszonych tematów.

- Uzyskanie warstwowego efektu cienia przy użyciu CSS i programu Photoshop.
- Tworzenie za pomocą CSS ultralekkich menu rozwijanych.
- Wstawianie dodatków Flasha bez łamania zasad zgodności ze standardami.

Efekt cienia

Jedną z najbardziej wpadających w oko cech witryny PGA było prawie trójwymiarowe podzielenie treści na warstwy. Efekt był subtelny, ale po dokładnym przyjrzeniu się stronie głównej (*www.pga.com/pgachampionship/2004/index.html*) widocznej na rysunku 3.1, można było zauważyć, że lewa kolumna wydaje się unosić nad otaczającą ją treścią. Nie był to tylko wizualny trick. Miało to także inne znaczenie. Lewa kolumna zawierała najnowsze wiadomości z mistrzostw. Wyniesienie tego obszaru nieco do góry (w niezauważalny sposób) sprawiło, że użytkownicy mogli szybko znaleźć miejsce, w którym znajdowały się najświeższe wiadomości.



Rysunek 3.1. Strona główna mistrzostw PGA

Lewa kolumna nie mogła być po prostu wyższa od sąsiednich, ale musiała częściowo zasłaniać treść, dzięki czemu wydawało się, że rzeczywiście się unosi nad stroną. Technikę tę zastosowano przy tworzeniu nagłówka *Top Story* w górnej części lewej kolumny. Ponieważ sprawiał wrażenie, jakby był złączony z kolumną i rzucał początkowy fragment cienia biegnącego w dół strony, *wyglądał* tak, jakby był utworzony w XHTML i unosił się nad filmem Flasha, ale to była tylko iluzja.

Efekt końcowy może wydawał się skomplikowany i wymagający dużej szybkości łączy, ale w rzeczywistości zajmował niewiele bajtów i kodu oraz można go było w miarę szybko utworzyć przy odrobinie zaawansowanego planowania. W rozdziale opisuję ten efekt uzyskany przy użyciu programu Photoshop, technologii Flash oraz CSS i XHTML.

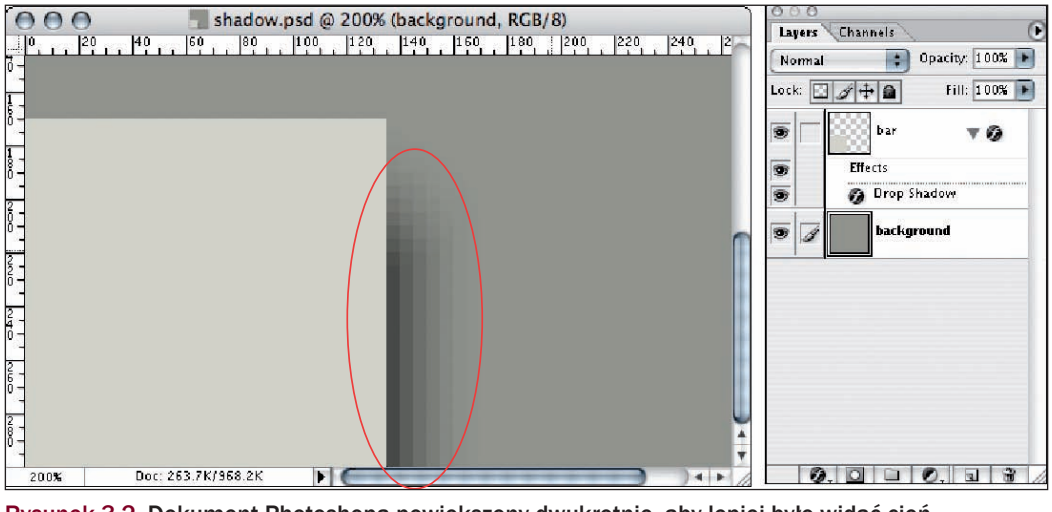
Tworzenie iluzji

Zanim przejdę do szczegółów, cofnę się na chwilę i jeszcze raz przedstawię cel — należy wizualnie scalić film Flasha umieszczony na górze strony ze znajdującą się pod nim treścią oraz stworzyć wrażenie, że lewa kolumna znajduje się na warstwie leżącej nad otaczającą ją treścią, poprzez utworzenie cienia biegnącego w dół. Trudność polega na tym, że efekt byłby zrujnowany, jeśli film Flasha lub elementy XHTML przesunęłyby się choćby o jeden piksel. Jednak precyzja CSS i XHTML sprawiła, że dało się to zrobić.

Filmy we Flashu, jak każdy wstawiany element, są prostokątne. Obiekty Flasha nie mogą mieć nieregularnych kształtów. Aby obejść te ograniczenia i utworzyć iluzję warstw, część lewej kolumny została przedstawiona w obiekcie Flasha. Film ten został wypozycjonowany z dokładnością do jednego piksela w taki sposób, że po dodaniu kodu XHTML obie części pasowały do siebie jak ulał, tworząc iluzję.

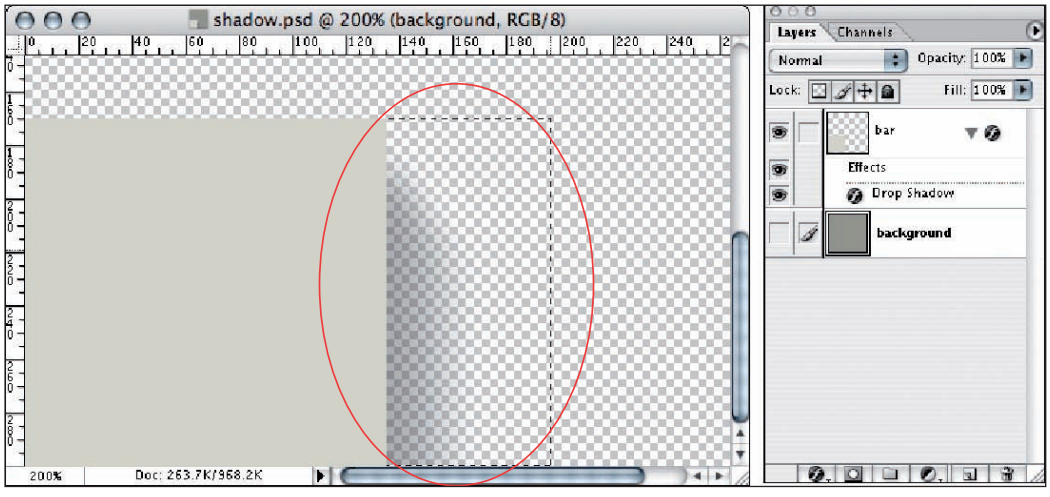
Najpierw w programie Photoshop powstał nowy dokument, który został wypełniony kolorem użytym w filmie Flasha i kolumnie biegnącej w dół strony. Utworzono też nową warstwę o nazwie *bar*. Następnie zaznaczono obszar o takiej samej szerokości jak lewa kolumna (przy użyciu narzędzia prostokątnego zaznaczania) i wypełniono go wybranym kolorem. Efekt cienia został zastosowany do warstwy *bar*, a sam cień został tak opracowany, aby nie był zbyt ciężki, ale na tyle widoczny, by stwarzać złudzenie oddzielenia od reszty treści.

Po ukończeniu dokument wyglądał tak, jak na rysunku 3.2.



Rysunek 3.2. Dokument Photoshopa powiększony dwukrotnie, aby lepiej było widać cień

Gdy cień był już dobry, wyłączono warstwę w tle i za pomocą narzędzia prostokątnego zaznaczania zaznaczono sam cień (rysunek 3.3). Skopiowano go do schowka i utworzono nowy dokument Photoshopa z przezroczystym tłem, do którego wklejono cień. Powstałą grafikę zapisano w formacie PNG i umieszczono w filmie Flasha (w którym znajdowała się już wektorowa figura o tej samej szerokości i wysokości).

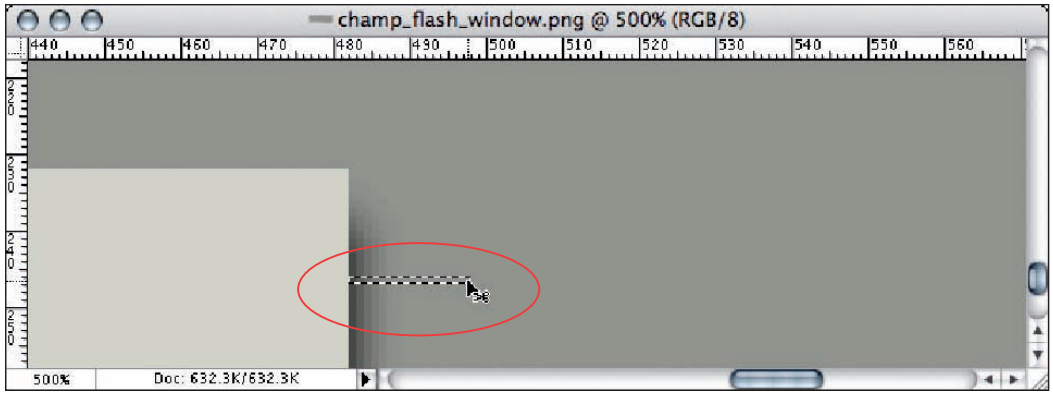


Rysunek 3.3. Zaznaczenie obszaru cienia

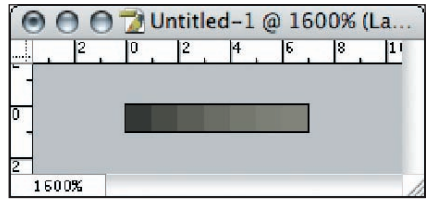
Możesz się zastanawiać, dlaczego zdecydowano się na format PNG zamiast popularniejszego GIF lub JPEG. Ten pierwszy jest preferowany podczas importu map bitowych do Flasha. Dlaczego? Ponieważ zawiera kanał alfa oferujący 256 poziomów przezroczystości. Dzięki temu można eksportować szczegółowe grafiki z edytorów map bitowych (Photoshop, Fireworks itp.), importować je do Flasha i publikować filmy zachowujące ich przezroczyste obszary przy jednoczesnym zastosowaniu stratnej kompresji JPEG. Jest to połączenie najlepszych cech — wizualna jakość formatu PNG i niewielkie rozmiary formatu JPEG.

Flash przestał już być potrzebny, teraz należało powielić cień przy użyciu CSS i XHTML. Aby cień nie biegł do samego dołu strony, trzeba było zbudować dla niego osobną sekcję. To wymagało utworzenia grafiki, która nie tworzyła żadnych dziur, przerw oraz innych anomalii i można było powtarzać ją w pionie. W związku z tym zaznaczono obszar o wysokości jednego piksela (rysunek 3.4) i odpowiedniej szerokości, aby zahaczyć także kawałek tła.

Po skopiowaniu tego obszaru utworzono nowy dokument (którego rozmiary zostały automatycznie odpowiednio ustawione przez Photoshop) i wklejono do niego skopiowany fragment (rysunek 3.5).

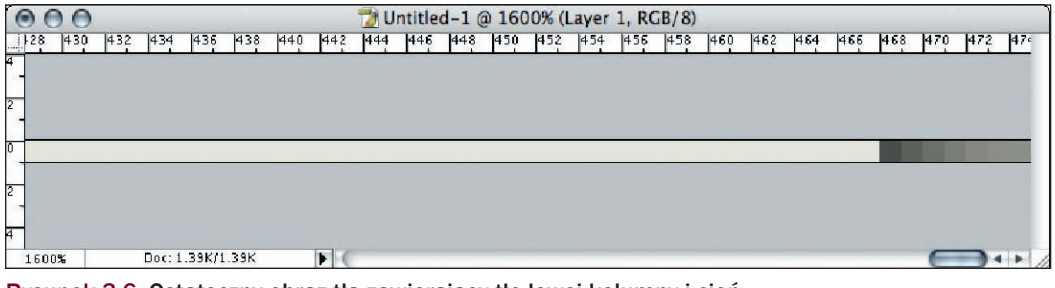


Rysunek 3.4. Przy użyciu narzędzia zaznaczania prostokątnego zaznaczono powtarzalny obszar o wysokości jednego piksela i szerokości pozwalającej zahaczyć kawałek tła



Rysunek 3.5. Zawartość schowka wklejona do nowego dokumentu i powiększona, aby ukazać szczegóły

W ten sposób poradzono sobie z cieniem, ale jeszcze brakowało jaśniejszego tła pod lewą kolumną. W tym celu zwiększono rozmiar cienia obrazu na odpowiednią szerokość, a sam cień wyrównano do prawej. Jak widać na rysunku 3.6, przezroczysty obszar został wypełniony kolorem, a całość zapisano jako obraz GIF.



Rysunek 3.6. Ostateczny obraz tła zawierający tło lewej kolumny i cień

Zatrzymam się na chwilę i dodam kilka słów na temat formatów plików. Czemu tutaj został użyty GIF zamiast PNG? Powodem jest niesławna przeglądarka Internet Explorer i jej słaba obsługa formatu PNG. Prawie osiem lat temu Microsoft rozgłaszał w białej księdze, że Internet Explorer 4.0 będzie „natywnie obsługiwać" ten format, w przeciwieństwie do „innych przeglądarek, które dodają obsługę formatu PNG jako osobny dodatek". Obietnic tych jednak nie spełniono, co więcej, musiało upłynąć ponad dziewięć lat — do pojawienia się Internet Explorera 7 — aby firma to zrobiła. Mimo że Internet Explorer 7 obsługuje już PNG, Internet Explorer 6 ma nadal bardzo wielu użytkowników, a więc w tym projekcie nie można było ich pominąć.

Na razie rozwiązanie przy użyciu obrazu GIF odpowiada większości przeglądarek oraz pozwala użyć bezstratnej kompresji idealnej dla obrazów niebędących zdjęciami.

Przyjrzyjmy się teraz kodowi CSS. W arkuszu stylów został utworzony kontener przy użyciu elementu `div` — niewidocznego bloku stanowiącego zewnętrzną ramkę służącą do pozycjonowania grup treści (czyli w tym przypadku lewej i prawej kolumny). Gdy zwiększała się ilość treści w kolumnach, otaczający je element odpowiednio powiększał swoje rozmiary. Dlatego właśnie tło zostało nadane kontenerowi, a nie samym kolumnom — wówczas obraz byłby powtarzany bez względu na ilość wewnętrznej treści. Załatwiono to przy użyciu poniższej reguły CSS:

```
#colwrap {
  width:740px;
  background:#96968C
url(http://i.pga.com/pga/images/pgachampionship/img/bg_home_content.gif) repeat-y;
}
```

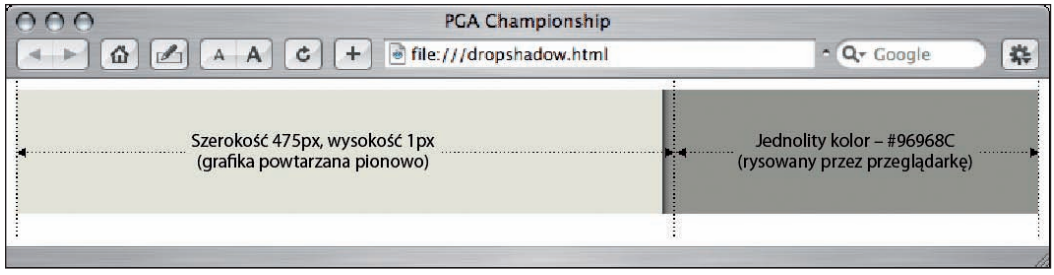
Najpierw ustawiono szerokość tego elementu, potem zdefiniowano atrybuty tła (najważniejszego w tym miejscu). Został ustawiony kolor taki sam jak filmu Flasha i prawej kolumny, podano adres URL wymaganego obrazu oraz poinstruowano przeglądarkę, żeby powtarzała ten obraz pionowo (`repeat-y`) przez całą długość elementu `div`.

Gdyby nie własność `repeat-y`, przeglądarka domyślnie powtórzyłaby obraz nie tylko w pionie, ale i *w poziomie*. To spowodowałoby, że lewa krawędź grafiki pojawiłaby się ponownie obok cienia. Oczywiście, to nie byłoby dobre dla projektu, dlatego postanowiono użyć własności `repeat-y`, aby powtórzyć obraz tyko w pionie. Gdyby potrzebne było powtarzanie tylko w poziomie, należałoby skorzystać z własności `repeat-x`.

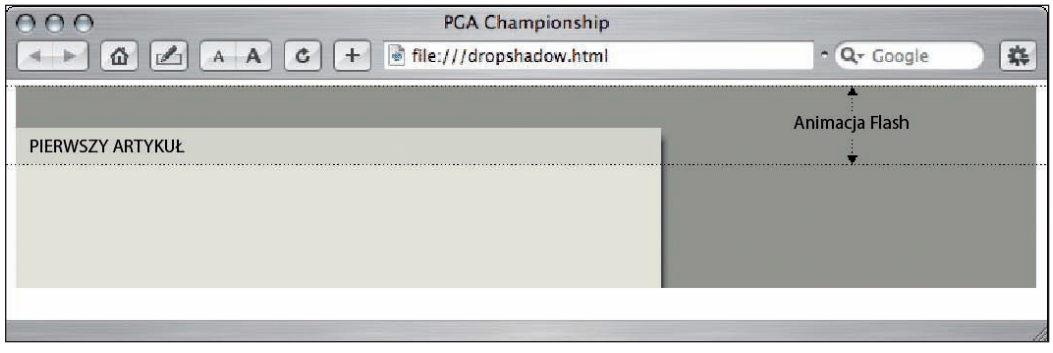
Ponieważ CSS pozwala na zastosowanie zarówno koloru, jak i obrazu do jednego elementu, można oszczędzić kilka bajtów poprzez wypełnienie obszarów jednolitego koloru przy użyciu CSS. Dlatego właśnie obraz tła kontenera został przycięty zaraz za cieniem. Resztę tła (ciemnoszary obszar, w który przechodzi cień) narysowała przeglądarka przy użyciu wyznaczonego koloru. Mimo że metoda ta nie była konieczna (całość można było umieścić na obrazku GIF), pozwoliła zaoszczędzić kilka bajtów. A wiadomo, że każdy bajt się przyda.

Ukończony element `div` wyglądał tak, jak na rysunku 3.7.

Po ukończeniu tła elementu `div` umieszczono nad nim film Flasha, który doskonale do niego pasował (rysunek 3.8).



Rysunek 3.7. Kontener div z grafiką w tle widziany w przeglądarce. Zostały dodane wskazówki informujące, który obszar jest zajmowany przez obraz, a który został narysowany przez przeglądarkę



Rysunek 3.8. Film Flasha umieszczony nad kontenerem z rysunku 3.7

Zanim przejdę dalej, podam jedną wskazówkę. Zawczasu upewnij się, że projekt jest na stałe przymocowany do swojego miejsca, najlepiej przy użyciu siatki elementów ustawionych z dokładnością do jednego piksela. Uzyskanie takiego efektu wymaga, by z góry znać ułożenie treści na stronie. Jeśli spróbujesz wyprzedzić czas i zastosować takie efekty do jeszcze nieustalonego układu, będziesz musiał więcej niż raz zaczynać wszystko od początku. Efekt, taki jak ten, potraktuj jak dopieszczanie projektu. Odrobina cierpliwości i planowania pozwoli uniknąć problemów i zaoszczędzi dużo czasu.

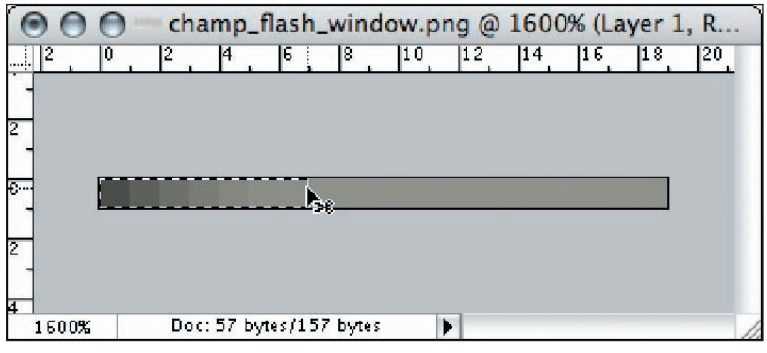
Dodanie realizmu

To jest dobre miejsce na podanie dodatkowej wskazówki, która umożliwi dodanie większego realizmu do efektu cienia.

W realnym świecie (tym poza monitorem) przezroczystość cienia jest zależna od powierzchni, na jaką pada. Jeśli jest biała, cień jest jasnoszary. Jeśli jest ciemna, cień wydaje się dużo ciemniejszy. To wszystko jest oczywiste, ale w płaskim świecie ekranów monitorów nie jest takie proste.

Na witrynie PGA posunięto się nieco dalej i ustawiono różne kolory cienia, zależne od tego, z jakim kolorem tła oddziaływał. Większość treści w prawej kolumnie miała przezroczyste tło, a więc korzystała z koloru zdefiniowanego pod spodem, ale odnośniki do wiadomości sprawiały problemy. Użytkownikom trudno było je odróżnić. W związku z tym, tło co drugiego elementu wiadomości było przyciemnione (jak wiersze w tabeli) i zmieniano odcień cienia, jeśli było trzeba.

W tym celu ponownie otwarto oryginalny dokument Photoshopa i zmieniono kolor na nieco ciemniejszy (#828279), który dobrze współgrał z resztą układu. Zaznaczono obszar o wysokości jednego piksela (tak jak poprzednio) i skopiowano ten obszar do nowego dokumentu. Grafika wyglądała bardzo podobnie do poprzedniej, tylko była nieco ciemniejsza (rysunek 3.9).



Rysunek 3.9. Cień z nieco ciemniejszym tłem zaznaczony narzędziem do prostokątnego zaznaczania

W kodzie XHTML utworzono listę nieuporządkowaną. Z semantycznego punktu widzenia elementy listy doskonale nadają się do oznaczania danych, takich jak paski nawigacyjne (o czym przekonasz się nieco dalej). W połączeniu z CSS można przy ich użyciu uzyskać wiele efektów wizualnych.

W prawej kolumnie szablonu XHTML została utworzona poniższa lista elementów wiadomości:

```
<ul class="stories">
  <li>DiMarco and Riley play their way into Ryder Cup</li>
  <li>'No question' PGA will return to Whistling Straits</li>
  <li>Sullivan lowest club professional at PGA since 1969</li>
  <li>PGA of America adjusts Sunday yardages at Whistling Straits</li>
</ul>
```

Następnie klasę przypisaną elementowi tej listy zdefiniowano w CSS:

```
ul.stories {
  margin:0;
  padding:0;
  color:#E9E9DF;
}
```

Najpierw przypisano klasę `stories` do elementu listy nieuporządkowanej. Dzięki temu zostały wyzerowane domyślne własności, które przeglądarka zastosowałaby wobec tego elementu. Ustawiono margines i dopełnienie na zero oraz zdefiniowano kolor tekstu w całej liście.

> ***Z technicznego punktu widzenia, można było opuścić element `ul` w selektorze i zostawić samą klasę. Mimo że przypisywanie klas bezpośrednio do elementów HTML nie jest najlepszym sposobem projektowania, tak napisane arkusze stylów znacznie łatwiej się czyta. Traktuj te elementy jak komentarze pozwalające szybko się zorientować, jakie jest przeznaczenie danego selektora po kilku miesiącach, gdy trzeba wprowadzić jakieś poprawki lub nad stroną pracuje więcej niż jedna osoba. Dobra organizacja na początku skutkuje oszczędnością czasu w przyszłości.***

Po sformatowaniu samego elementu `ul` przyszedł czas na zajęcie się elementami wewnątrz listy:

```
ul.stories li {
  list-style:none;
  margin-bottom:2px;
  padding:4px 4px 4px 10px;
}
```

Przeanalizuję ten kod wiersz po wierszu. Najpierw własność `list-style` została ustawiona na `none`, co spowodowało zniknięcie domyślnych okrągłych punktorów. Następnie dodano niewielki margines dolny, aby odsunąć od siebie elementy listy w pionie, oraz dopełnienie (`4` piksele od góry, prawej i dołu oraz `10` z lewej strony).

Domyślnie wartości te zostały zastosowane do każdego elementu listy nieuporządkowanej należącej do klasy `stories`. Na tym etapie wszystkie miały takie samo tło (prześwitujące spod spodu), ale poniższa reguła odpowiada za dodatkowy efekt:

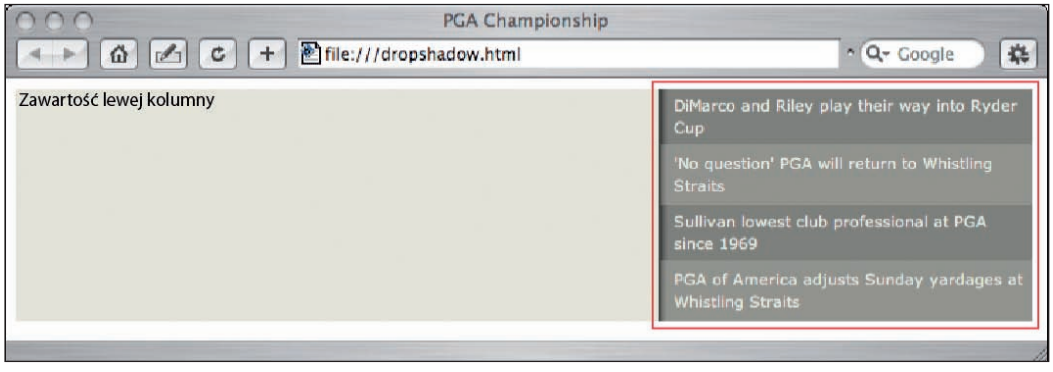
```
ul.stories li.odd {
  background:#828279
url(http://i.pga.com/pga/images/pgachampionship/img/bg_stories_shadow.gif) repeat-y;
}
```

Dzięki pięknu dziedziczenia klasa `odd` nie musiała już implementować wszystkich wcześniejszych własności, a tylko tę, która była potrzebna — tło. Zostało zastosowane ciemniejsze tło i podano adres URL drugiego obrazu tła oraz poinstruowano przeglądarkę, aby powtórzyła obraz tylko w pionie.

Kod listy nieuporządkowanej został dodany do dokumentu XHTML i klasa `odd` została zastosowana (ręcznie, chociaż można było to zrobić przy użyciu JavaScriptu lub języka PHP itp.) do co drugiego elementu listy:

```
<ul class="stories">
  <li class="odd">DiMarco and Riley play their way into Ryder Cup</li>
  <li>'No question' PGA will return to Whistling Straits</li>
  <li class="odd">Sullivan lowest club professional at PGA since 1969</li>
  <li>PGA of America adjusts Sunday yardages at Whistling Straits</li>
</ul>
```

Wydaje się, że powyższa lista nieuporządkowana jest częścią prawej kolumny umieszczoną pod cieniem głównego obszaru treści, ale w rzeczywistości znajduje się *nad* utworzonym wcześniej tłem (rysunek 3.10). Sztuczka polega na umieszczeniu prawej kolumny (zawierającej listę nieuporządkowaną) bezpośrednio nad prawą krawędzią lewej kolumny. To tworzy złudzenie, że ciemniejsze tło elementów listy jest częścią cienia, podczas gdy w rzeczywistości znajduje się nad nim.



Rysunek 3.10. Elementy listy z ciemniejszym tłem umieszczone w kontenerze div

Oto kod CSS prawej i lewej kolumny:

```
#lcol {
  width:468px;
  float:left;
}
#rcol {
  width:271px;
  float:right;
}
```

Podstawowy kod XHTML potrzebny do uzyskania tego efektu wygląda następująco:

```
<div id="colwrap">
  <div id="lcol">
    <! – Zawartość lewej kolumny – >
  </div>
  <div id="rcol">
    <ul class="stories">
      <li class="odd">DiMarco and Riley play their way into Ryder Cup</li>
      <li>'No question' PGA will return to Whistling Straits</li>
      <li class="odd">Sullivan lowest club professional at PGA since 1969</li>
      <li>PGA of America adjusts Sunday yardages at Whistling Straits</li>
    </ul>
  </div>
</div>
```

Dodatkowy efekt cienia został ukończony.

Jeśli miałbyś zapamiętać tylko jedną rzecz, pamiętaj o tym, że — wykorzystując zdolność przeglądarki do automatycznego powtarzania obrazów tła i jednoczesnego stosowania koloru tła tego samego elementu — wizualną głębię do płaskiego i statycznego projektu można dodawać na wiele sposobów. Wymaga to tylko nieco cierpliwości, planowania i poeksperymentowania.

Tworzenie w CSS menu rozwijanych

Pod koniec lat 90. ubiegłego wieku, w najlepszych czasach dotcomów znakiem rozpoznawczym wyrafinowanej, utworzonej przy użyciu najnowocześniejszych technik witryny było rozwijane menu nawigacyjne. Menu takie prezentowały się bardzo atrakcyjnie, ale za nimi stało mnóstwo kodu JavaScript, rozdętego HTML, a nawet elementów działających tylko w konkretnych przeglądarkach. Zamiast podnosić komfort użytkownika wiele wczesnych menu rozwijanych tylko go pogarszało (szczególnie wtedy, gdy nie chciały działać), a na dodatek niepotrzebnie rozdymało objętość kodu.

Wówczas pojawiły się CSS i magiczny selektor pseudoklasy `:hover`. Guru CSS, tacy jak Eric Meyer i Christopher Schmitt, publikowali kursy uczące, jak wykorzystywać możliwości nowej funkcji. Stosując ją w połączeniu ze zwykłymi listami, można było uzyskać menu rozwijane podobne do tworzonych przy użyciu JavaScriptu, ale dużo lżejsze i mniej skomplikowane.

Było jednak jedno „ale”: Internet Explorer dla Windows. Zdecydowanie najpopularniejsza przeglądarka internetowa nie obsługiwała w pełni pseudoklasy `:hover` (ani CSS, ale to już inna historia), przez co nie radziła sobie z menu rozwijanymi w CSS. Menu te stały się co najwyżej zajęciem dla hobbystów.

Zmiany zaczęły się w 2003 roku, kiedy Patrick Griffiths i Dan Webb postawili społeczność CSS na głowie swoim skryptem *Suckerfish Dropdowns* (*www.alistapart.com/articles/dropdowns*). Był to lekki, zbudowany przy użyciu CSS system menu rozwijanych, który działał w prawie każdej przeglądarce, włącznie z Internet Explorerem dla Windows. Suckerfish był rewelacją nie tylko pod względem objętości, lecz także zgodności z przeglądarkami, standardami; był bogaty semantycznie oraz szeroko dostępny.

Dodatkowo menu te były niezwykle łatwe do zbudowania. Każdy, kto potrafił utworzyć listę nieuporządkowaną w XHTML, miał ułatwione zadanie. Wszystkie właściwości prezentacyjne i funkcjonalne były kontrolowane przez niewielki zestaw reguł stylistycznych.

Kilka miesięcy później Griffiths i Webb zaktualizowali swój produkt, nazywając go *Son of Suckerfish*. Była to jeszcze lżejsza i lepiej współpracująca z przeglądarkami wersja poprzedniego skryptu, umożliwiająca tworzenie wielopoziomowych menu. W witrynie PGA użyto właśnie drugiej wersji, ale nie będę tu zgłębiać podstawowych szczegółów (przykłady można pobrać ze strony *www.htmldog.com/articles/suckerfish*). Opiszę natomiast, jakich dokonano przeróbek, omówię potencjalne pułapki oraz podam ogólne wskazówki dotyczące użytkowania tych menu.

Pozycjonowanie menu rozwijanych

Pierwszy problem napotkany podczas modyfikowania skryptu Suckerfish na potrzeby witryny PGA dotyczył pozycjonowania menu. Domyślnie zagnieżdżone wysuwane menu pokazywały się bezpośrednio pod swoim elementem rodzicem, w zależności od wysokości grafiki lub tekstu, które się w nim znajdowały. W witrynie PGA grafiki rodziców były krótsze niż otaczający je obszar (aby zmniejszyć wagę plików). W związku z tym, menu rozwijane, zamiast pojawiać się pod białym paskiem nawigacyjnym (tak jak w ostatecznej wersji), pojawiały się pod odnośnikami graficznymi.

Łącza graficzne można było edytować z dodatkową pustą przestrzenią na dole, aby przenieść menu rozwijane do pożądanej lokalizacji, ale to spowodowałoby zwiększenie rozmiaru plików i powstanie łączy w miejscach, w których nic nie widać. Wyzwanie zatem polegało na tym, aby zepchnąć menu rozwijane pod biały pasek nawigacyjny, nie naruszając istniejącej treści.

Pierwszy krok był łatwy. Każda zagnieżdżona nieuporządkowana lista (menu rozwijane) była pozycjonowana bezwzględnie, a więc przeniesiono je niżej przy użyciu własności `top`:

```
#nav li ul {
  position:absolute;
  left:-999em;
  top:20px;
}
```

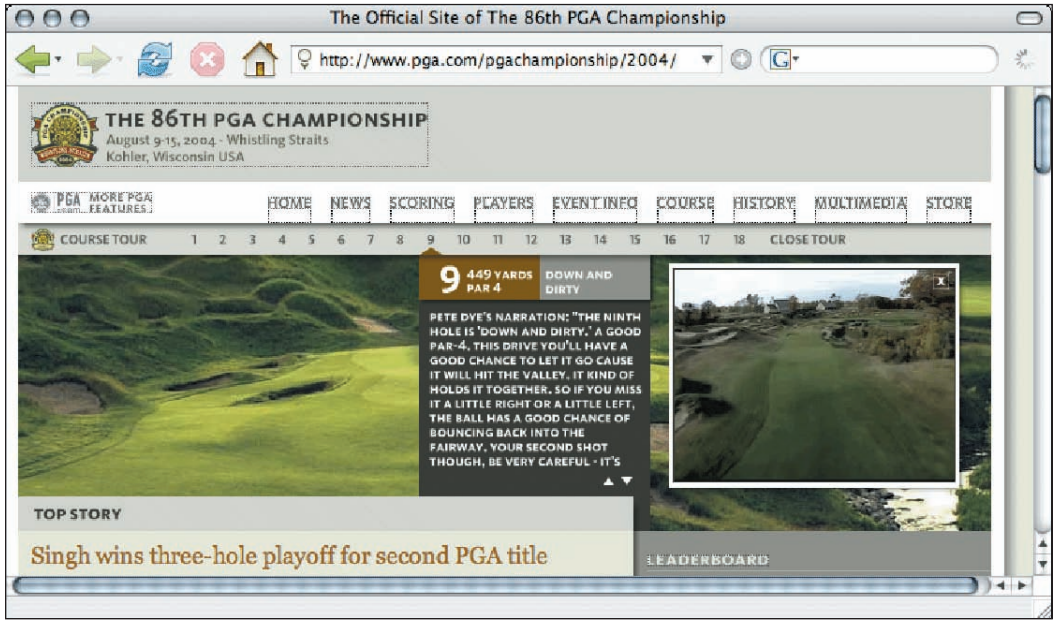
W ten sposób przesunięto każde menu rozwijane o 20 pikseli poniżej jego rodzica, a zatem pod biały pasek nawigacyjny. Pojawił się jednak nowy problem. Obszar między głównymi odnośnikami a menu rozwijanymi (ten sam, który nie powinien być odnośnikiem) wyłączał efekt rollover, gdy kursor przesuwał się w dół. Zatem kolejnym krokiem było podtrzymanie widoczności menu rozwijanych, gdy kursor wchodził na ten pusty obszar.

Domyślnie wysokość elementu listy jest zdeterminowana przez jego treść. Można to jednak zmienić za pomocą CSS:

```
#nav li {
  position:relative;
  float:left;
  margin:0 15px 0 0;
  padding:0;
  width:auto;
  height:20px;
}
```

Najważniejsza w tym kodzie jest własność `height`. Jej zdefiniowanie sprawiło, że przesłonięto domyślne, wspomniane wcześniej zachowanie elementów listy i niewidoczny blok każdego z nich rozciągnął się, zapełniając lukę. Teraz elementy listy zachowują się tak, jakby zawierały obrazy o wysokości 20 pikseli, ale w rzeczywistości są znacznie krótsze. Jednak przeglądarki nie widzą różnicy, dzięki czemu menu działają zgodnie z oczekiwaniami.

Aby zobaczyć efekt, spójrz na rysunek 3.11. Za pomocą dostępnego bezpłatnie rozszerzenia Web Developer (którego autorem jest Chris Pederick — *http://chrispederick.com/work/web-developer*) dla Firefoksa sprawiłem, że niewidoczne bloki elementów listy stały się widzialne. Stanowi to wizualne potwierdzenie dokonanych zmian i pozwala zobaczyć, co w rzeczywistości wyświetla przeglądarka. Dodatek ten przydał się wiele razy podczas pracy nad witryną PGA i polecam go każdemu, kto zajmuje się tworzeniem stron internetowych.



Rysunek 3.11. Strona PGA w Firefoksie z włączonym obramowaniem elementów listy i obrazów przy użyciu dodatku Web Developer

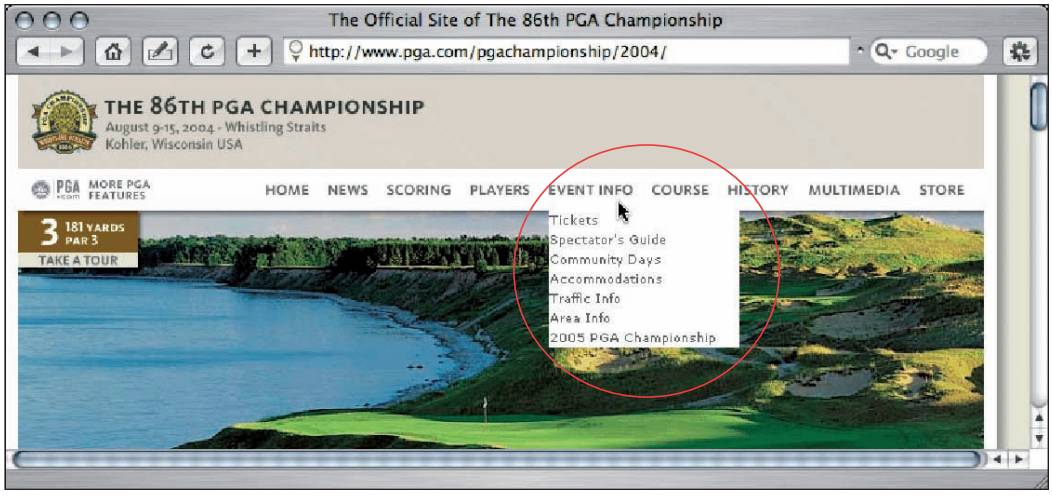
Dostrajanie: stylizacja menu rozwijanych

Po zmuszeniu menu rozwijanych do prawidłowego funkcjonowania i pojawiania się we właściwym miejscu nadszedł czas na modyfikację wyglądu ich opcji.

Najpierw kolor tła zagnieżdżonych list nieuporządkowanych ustawiono na biały oraz nadano im jednakową szerokość (równą najdłuższemu elementowi):

```
#nav li ul {
  margin:0;
  padding:0;
  position:absolute;
  left:-999em;
  top:20px;
  background:#fff;
  width:146px;
}
```

Jak widać na rysunku 3.12, problem łatwo dostrzec. Lewa krawędź każdego menu rozwijanego jest wyrównana z lewą krawędzią zawierającego je elementu listy, a opcje nie są wystarczająco oddzielone.



Rysunek 3.12. Źle wyrównane menu z mdło wyglądającymi opcjami

W związku z tym, dokonano pewnych modyfikacji:

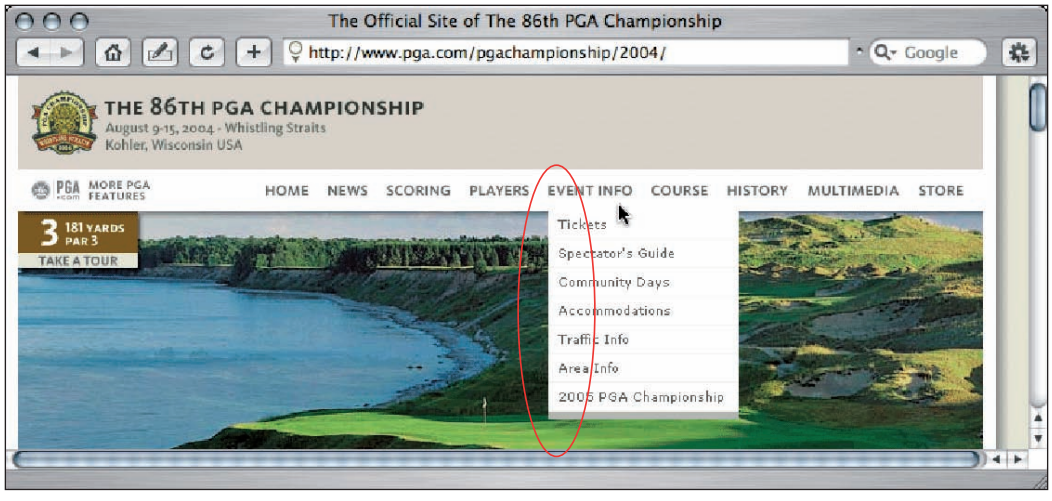
```
#nav li li {
  height:auto;
  margin:0;
  padding:0;
  width:100%;
  font-size:9px;
  border-bottom:1px solid #F5F5F0;
}
```

Ponieważ szerokość każdego zagnieżdżonego elementu listy została ustawiona na 100%, rozszerza się on na całą szerokość swojego elementu nadrzędnego — który w tym przypadku ma szerokość 146 pikseli. Jeśli domyślna szerokość elementów listy nie jest zmieniona, przeglądarka rysuje dolną krawędź tylko na szerokość treści. Ustawienie jej na 100% nadaje opcjom jednolity wygląd, bez względu na treść.

Następnie zajęto się zawartością tekstową:

```
#nav li li span {
  display:block;
  margin:0;
  padding:3px 4px 3px 7px;
  position:relative;
}
```

Aby lepiej można było kontrolować pozycję każdego bloku tekstu, opcje zostały opakowane w znacznik `span`, który pozwala uniknąć używania kolejnej klasy i lepiej nadaje się z semantycznego punktu widzenia niż `div`, `p` lub jakikolwiek inny. A więc własność `display` elementu `span` została zmieniona na `block` (domyślnie jest `inline`). Dzięki temu można definiować atrybuty bloków, takie jak `margin` i `padding`. Po odpowiednim zdefiniowaniu dopełnienia menu wygląda tak, jak na rysunku 3.13.

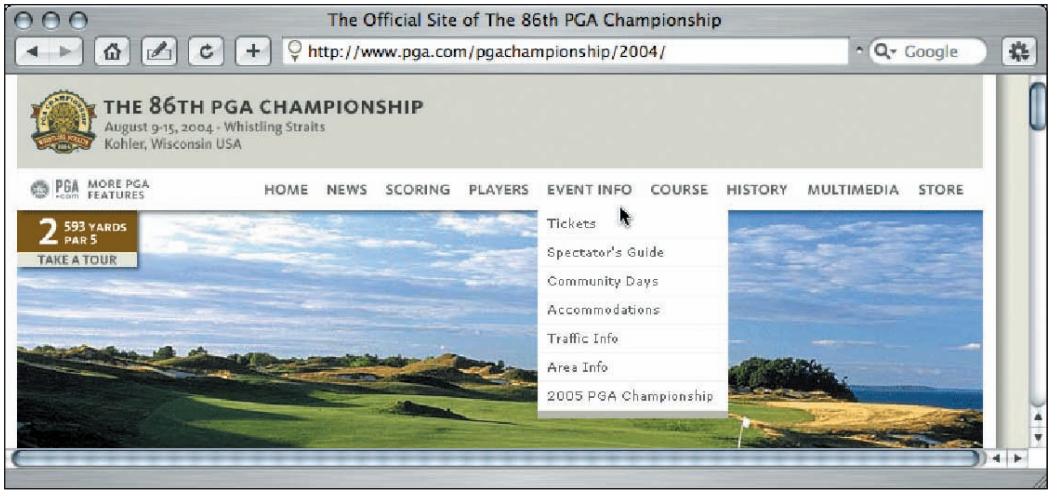


Rysunek 3.13. Opcje menu rozwijanego po sformatowaniu, ale niewyrównane z lewym marginesem głównych odnośników nawigacyjnych

Menu wyglądają na ukończone, ale do zrobienia pozostała jeszcze jedna rzecz. Opcje tekstowe nie są wyrównane z lewą krawędzią swoich elementów nadrzędnych. Mimo że nie trzeba tego zmieniać, zdecydowano, że będą lepiej wyglądały, jeśli zostaną przesunięte nieco w lewo. Na szczęście, wystarczy tylko przesunąć każdą listę nieuporządkowaną w lewo:

```
#nav li ul {
  margin:0 0 0 -8px;
  padding:0;
  position:absolute;
  left:-999em;
  top:20px;
  background:#fff;
  width:146px;
}
```

Dobrze, że można używać ujemnych wartości! Zmiana lewego marginesu na `-8` (kolejność wartości marginesów to góra, prawa, dół, lewa) spowodowała przeniesienie nieuporządkowanej listy o osiem pikseli w lewo (gdyby wartość była dodatnia, przeniesienie byłoby w prawo). Dzięki temu lewa krawędź każdej opcji tekstowej wyrównuje się ze swoim nadrzędnym elementem tak, jak na rysunku 3.14.



Rysunek 3.14. Ukończone menu rozwijane

Uwaga na pułapkę

Po zapoznaniu się z technikami modyfikowania oryginalnego kodu źródłowego przyszedł czas na ostrzeżenie przed potencjalnym problemem, który może pojawić się podczas używania skryptu Suckerfish. *Wymaga on włączonego JavaScriptu w Internet Explorerze dla Windows*. Zalicz to do zestawu nieudolnych implementacji CSS w tej przeglądarce. W przeglądarkach Mozilla, Firefox i Safari skrypt ten działa wyłącznie przy użyciu CSS, ale w Internet Explorerze do działania wymaga JavaScriptu. Hack ten jest niewielki i naprawia brak obsługi pseudoklasy `:hover`, dodając niestandardowe zachowania do obiektowego modelu dokumentu (DOM). W tym leży sedno problemu. Jeśli użytkownik Internet Explorera wyłączy JavaScript, menu w ogóle się nie pojawi. Mimo że bardzo mało osób tak postępuje, trzeba to brać pod uwagę.

Użyteczność menu rozwijanych

Problem zgodności z przeglądarkami został rozwiązany, zatem poniżej przedstawiam kilka wskazówek dotyczących wdrażania menu rozwijanych na własnych stronach.

- **Wskazówka 1. Zagwarantuj wyjście awaryjne.** Ze względu na opisane wcześniej problemy ze zgodnością, należy bezwzględnie wyposażyć stronę w dodatkowy system nawigacji, na wypadek gdyby główny nie zadziałał. Inaczej odwiedzający nie będą mogli poruszać się po witrynie. Powinna to być standardowa procedura podczas używania każdego rodzaju nawigacji opartej na menu rozwijanych, nie tylko Suckerfish.
- **Wskazówka 2. Uważaj na Internet Explorera, Suckerfisha i Flasha.** Przeglądarka Internet Explorer dla Windows, napotykając film Flasha, przenosi go na sam wierzch stosu warstw. Oznacza to, że menu mogą znaleźć się pod filmem, co uniemożliwi użytkownikom klikanie jego elementów. Rozwiązaniem jest użycie parametru `wmode` (szczegółowe informacje na ten temat znajdują się na stronie *www.macromedia.com/*
↪support/flash/ts/documents/flash_top_layer.htm).
- **Wskazówka 3. Jeśli dokument zawiera warstwy, używaj własności `z-index`.** Jeżeli strona zawiera obiekty na warstwach z własnością `z-index`, nawigacja Suckerfish również musi ją mieć, ale ustawioną na poziom wyższy niż wszystkie pozostałe.

Własność tę można zastosować wobec nadrzędnej listy nieuporządkowanej lub (tak zrobiono na stronie PGA) można nawigację wstawić do elementu `div` i własność `z-index` zastosować do niego. Wtedy nawigacja będzie znajdowała się nad wszystkim i po rozwinięciu menu zawsze będzie na wierzchu.

Końcowe uwagi

Skoro jest tyle problemów, po co w ogóle używać nawigacji Suckerfish? Odpowiedź jest prosta. Poza opisanymi niedogodnościami, Suckerfish jest jednak najlepiej dostępnym, działającym we wszystkich przeglądarkach rozwiązaniem. Ponadto zajmuje znacznie mniej bajtów niż jakiekolwiek inne konkurencyjne rozwiązanie oraz jest łatwiejszy w utrzymaniu i modyfikacji. Jeśli implementujesz menu w witrynie z dużym ruchem (witryna PGA np. odnotowywała miliony odwiedzin na godzinę), zastosowanie lekkiego wariantu jest optymalnym rozwiązaniem.

Wstawianie filmów Flasha w zgodzie ze standardami

Jednym z najczęstszych problemów nękających twórców stron internetowych dbających o zgodność ze standardami jest wstawianie animacji Flasha. Większość z nich kopiuje i wkleja standardowy kod wygenerowany przez Flash. Ponieważ jednak jest on naleźony różnymi niestandardowymi atrybutami i elementami, obraca wniwecz zgodność dokumentu ze standardami sieciowymi. Znacznik `embed` nie należy do żadnej specyfikacji W3C.

Na szczęście, można te problemy obejść. Poniżej przedstawiam trzy najpopularniejsze obecnie rozwiązania problemu zgodności ze standardami podczas wstawiania treści Flasha na strony internetowe.

Metoda Flash Satay

Metoda Flash Satay (*www.alistapart.com/articles/flashsatay*) usuwa znacznik `embed` i niektóre niepotrzebne atrybuty znacznika `object`. Działa doskonale, ale ma jedną dużą wadę: w przeglądarce Internet Explorer dla Windows animacje Flasha zaczynają działać dopiero po załadowaniu całego filmu.

Metoda Satay oferuje rozwiązanie tego problemu, polegające na oszukiwaniu Internet Explorera poprzez dodanie pustego filmu kontenera z takimi samymi parametrami (szerokość, wysokość itd.) jak prawdziwy i załadowanie pożądanej treści za pomocą tego filmu zastępczego. Internet Explorer wówczas z powodzeniem odtwarza animację strumieniowo, a kod XHTML jest poprawny. Koszt to konieczność tworzenia dodatkowych pustych filmów i katalogów.

Wpisywanie znaczników object i embed przy użyciu JavaScriptu

Metoda ta polega na pozostawieniu bez zmian elementów `object` i `embed`, ale przeniesieniu ich do zewnętrznego pliku JavaScript. Treść Flasha jest wówczas zapisywana w dokumencie za pomocą serii metod JavaScript `document.write`. Walidatory (świetny jest walidator W3C — *http://validator.w3.org*) widzą tylko poprawny element JavaScript — a nie znajdujące się w nim elementy `object` i `embed`, dzięki czemu wyniki walidacji są pozytywne.

To rozwiązanie zastosowano w witrynie PGA. Nie tylko kod XHTML pozostał poprawny, lecz również dzięki użyciu JavaScriptu można było sprawdzić obecność wtyczek w przeglądarce, na wypadek gdyby trzeba było załadować alternatywną treść.

Po utworzeniu pliku JavaScript (zbyt długi, aby go tutaj drukować — *http://www.pga.com/*
➥pgachampionship/2004/js/flash_home.js) należy go dołączyć do strony w następujący sposób:

```
<script type="text/javascript"
src="http://www.pga.com/pgachampionship/2004/js/flash_home.js"></script>
```

Metoda nie jest idealna. Po pierwsze, puryści powiedzą, że plik JavaScript jest w rzeczywistości koniem trojańskim przemycającym nieprawidłowy i nieobsługiwany kod XHTML w tajemnicy przed walidatorami (tak rzeczywiście jest). Po drugie, wstawiając treść za pomocą JavaScriptu, zakładasz, że użytkownicy go nie wyłączyli u siebie (większość osób tego nie robi, ale niektórzy wyłączają, aby przyspieszyć przeglądanie i pozbyć się reklam). W końcu każda animacja Flasha wymaga własnego osobnego pliku *.JS*, co nie jest dużym problemem przy małej liczbie filmów, ale szybko może stać się uciążliwe.

SWFObject

Metoda SWFObject opublikowana kilka miesięcy po premierze witryny PGA z 2004 roku jest najbardziej wyrafinowanym i niezawodnym, aktualnie dostępnym rozwiązaniem. Napisany w JavaScripcie przez Geoffa Stearnsa pakiet stanowi bezpośrednią odpowiedź na ograniczenia obu wymienionych wcześniej metod. Umożliwia stosowanie prostszego kodu przechodzącego walidację jako XHTML 1.0 Transitional i wyżej.

SWFObject oferuje wszystko, czego potrzebuje programista Flasha — wykrywanie odtwarzacza, możliwość zaprezentowania alternatywnej treści tym, którzy go nie mają, metody przekazywania dodatkowych parametrów i zmiennych, pozycjonowanie obiektów w dowolnym miejscu za pomocą elementów `div`, a nawet zmienną pozwalającą obejść wykrywanie odtwarzacza i wymuszenie wyświetlenia filmu bez względu na to, czy w przeglądarce został zainstalowany ten dodatek, czy nie.

SWFObject jest też przyjazna dla wyszukiwarek, co jest rzadkością we Flashu. Użytkownicy tworzą element `div` w swoich dokumentach, zapełniają go zwykłą treścią HTML, jaka później może zostać zindeksowana i wyświetlona tym, którzy nie mają odpowiedniej wtyczki. Jeśli wtyczka Flasha jest zainstalowana, zawartość elementu `div` zostaje zastąpiona animacją. Zatem zarówno ci, którzy mają wtyczkę, i jak ci, którzy jej nie mają, mogą oglądać bogatą treść strony, a programista nie musi się zbytnio napracować.

Więcej informacji o pliku SWFObject (który jest darmowy) można znaleźć na stronie *http://blog.deconcept.com/swfobject*.

Podsumowanie

Zaprezentowałem tu wiele informacji, od tworzenia wizualnych efektów w Photoshopie po menu rozwijane, pozycjonowanie elementów przy użyciu CSS i problemy z walidacją treści Flasha w dokumentach XHTML. Przedstawione tutaj techniki powinny zachęcić Cię do dalszych poszukiwań i eksperymentów przy użyciu CSS.

W następnym rozdziale przyjrzysz się procesowi przeprojektowania witryny University of Florida. Dowiesz się, jaka była jej historia, jakie problemy napotkano przy aktualizowaniu starej treści oraz szczegółowo zapoznasz się z użytym na niej kodem CSS.