CSS. Gotowe rozwiązania

Autor: Richard York
Tłumaczenie: Łukasz Piwko
ISBN: 83-246-0574-6
Tytuł oryginału: CSS Instant Results
Format: B5, stron: 392

Wykorzystaj w swoich projektach możliwości arkuszy stylów

- Stwórz efektowny i czytelny mechanizm nawigacji
- Użyj stylów w internetowym kliencie poczty
- Popraw wygląd formularzy na stronach WWW

Kaskadowe arkusze stylów zmieniły oblicze sieci WWW. Dzięki nim definiowanie wyglądu strony WWW stało się znacznie prostsze. Przed ich pojawieniem się wygląd każdego elementu strony był określany w kodzie HTML. Zmiana układu graficznego, kolorystyki lub kroju czcionki wymagała modyfikacji każdego ze znaczników. Dziś, za pomocą arkusza stylów, możemy przedefiniować wygląd całej witryny, zmieniając tylko jeden plik zawierający style. Jednak technologia CSS umożliwia nie tylko określanie kroju i wielkości czcionki – pozwala także na uzyskanie ciekawych efektów wizualnych i nadanie stronie niepowtarzalnego wyglądu.

Książka „CSS. Gotowe rozwiązania" to przegląd dziesięciu projektów witryn WWW opartych na kaskadowych arkuszach stylów. Czytając ją, poznasz różne zastosowania technologii CSS – od definiowania wyglądu elementów strony po tworzenie interfejsów użytkownika dla aplikacji internetowych. Znajdziesz tu przykłady atrakcyjnego i funkcjonalnego mechanizmu nawigacji, dynamicznych list rozwijanych i przeglądarki plików. Każdy z projektów został przedstawiony w takiej postaci, że implementacja go we własnych pracach nie sprawi Ci żadnego problemu.

- Nawigacja oparta na zakładkach
- Wielokolumnowy układ strony WWW
- Dynamiczne listy rozwijane
- Internetowy pokaz slajdów
- Zmiana wyglądu formularzy
- Interfejs użytkownika dla przeglądarki plików
- Kalendarz z terminarzem

Nadaj swoim stronom niepowtarzalny wygląd

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

Spis treści

1

Zakładki

Zakładki stały się wszechobecne na stronach internetowych — można je znaleźć prawie wszędzie (jednym z najlepszych przykładów może być strona *Apple.com*). Kiedyś strony internetowe tworzono głównie w oparciu o mapy odnośników lub proste obrazki i tabele, które, umieszczone wewnątrz znacznika `<a>`, służyły jako zakładki. W niniejszym rozdziale prezentuję kilka sposobów implementacji zakładek na stronie przy użyciu CSS. Takie podejście pozwala na utworzenie czytelniejszego i bardziej dostępnego kodu.

Mimo nielicznych problemów każda bardziej znacząca przeglądarka potrafi obsłużyć prosty, przyjazny dla użytkownika i stabilny system zakładek. Jak to zawsze bywa z CSS, nie każde podejście będzie działało bez zarzutów w każdej z nich. Aby temu zapobiec, opisuję obejścia i sztuczki niezbędne do tego, aby nasz projekt wyglądał bez zarzutów w jak największej liczbie przeglądarek. W dalszej części rozdziału pokażę, jak przy użyciu biblioteki open source JavaScript IE7 spowodować, aby przeglądarka Internet Explorer w wersjach 5.5 oraz 6 była kompatybilna z bardziej zaawansowanymi technikami, a także jak JavaScript pozwala na pokonanie przeszkód niekompatybilności pomiędzy przeglądarkami nie tylko przy użyciu biblioteki IE7, lecz także przy użyciu innych narzędzi, takich jak np. stworzony specjalnie dla Internet Explorera atrybut CSS `expression()`, który odgrywa niebagatelną rolę w wielu przedstawionych w niniejszej książce projektach.

Przed rozpoczęciem pracy nad projektem trzeba go dokładnie rozplanować. Należy zadać sobie pytanie: „Jakie właściwości powinien mieć mój system zakładek?”. Poniżej znajduje się lista najważniejszych celów projektu:

- Kod użyty do utworzenia zakładek musi być prosty i zrozumiały.
- Tekst użyty w zakładkach nie może znajdować się na obrazkach.
- Zakładki muszą zmieniać kolor, gdy użytkownik najeżdża na nie kursorem.
- Zakładka odpowiadająca aktualnie wczytanej stronie musi być wyróżniona.

Następny podrozdział opisuje komponenty CSS, XHTML oraz JavaScript potrzebne do osiągnięcia zamierzonych celów.

Projekt

Projekt prezentowany w niniejszym rozdziale nie jest zbyt atrakcyjny wizualnie. Moim celem było raczej pokazanie, jak w prosty sposób stworzyć system zakładek przy użyciu CSS bez obrazków i tabel. W dalszej części prezentuję, jakie należy wnieść poprawki, aby zakładki płynnie się rozciągały oraz w jaki sposób ustawić właściwości ramek oraz tła (chociaż wersja podstawowa projektu jest bardzo prosta). Poniżej znajduje się lista celów z poprzedniego podrozdziału wzbogacona o opis właściwości CSS i kodu XHTML potrzebnych do ich osiągnięcia:

- **Kod użyty do stworzenia zakładek musi być prosty i zrozumiały.** Pierwszy cel jest raczej prosty do osiągnięcia — mówi on, żeby dane umieszczać wewnątrz właściwych znaczników. Oznacza to, że struktura naszego systemu zakładek będzie oparta na nienumerowanej liście HTML. Ze względu na fakt, że to, co chcemy zaprezentować, to nic innego jak lista odnośników (niekoniecznie w określonej kolejności), elementy `<ul>` i `<li>` wydają się być najlepszym rozwiązaniem. Dzięki CSS możliwa jest zmiana listy nienumerowanej na zakładki. Dokonuje się tego poprzez usunięcie domyślnego formatowania elementów listy (atrybuty `margin`, `padding` oraz `list-style`). Następnie elementom `<li>` nadaje się własne właściwości ramek, marginesów oraz, aby elementy występowały jeden obok drugiego, używa się albo atrybutu CSS `float`, bądź stosuje się pozycjonowanie absolutne (deklaracja `position: absolute`). W niniejszym projekcie korzystamy z atrybutu `float`, ponieważ jego zastosowanie jest trochę mniej kłopotliwe niż pozycjonowanie absolutne.
- **Tekst użyty na zakładkach nie może znajdować się na obrazkach.** Drugi cel projektu związany jest z dostępnością i przyjaznością dla robotów wyszukujących. Jeżeli tekst zakładek znajdowałby się na obrazkach, to osoby niewidome korzystające np. z przeglądarki JAWS nie byłyby w stanie go odczytać. Dodatkowo także roboty wyszukujące nie odczytałyby takiego tekstu, który może przecież zawierać wartościowe z punktu widzenia pozycjonowania słowa kluczowe. W dalszej części rozdziału pokazuję sposób stosowania obrazów tła przy zachowaniu tekstu i utrzymaniu widzialności dla robotów oraz dostępności.
- **Zakładki muszą zmieniać kolor, gdy użytkownik najeżdża na nie kursorem.** Cel ten jest bardzo prosty do osiągnięcia. Chcemy po prostu, aby użytkownik miał wizualne potwierdzenie, że dana zakładka jest elementem nawigacji. Jednym ze sposobów osiągnięcia tego efektu jest zmiana koloru tła za pomocą pseudoklasy CSS `:hover`.
- **Zakładka odpowiadająca aktualnie wczytanej stronie musi być wyróżniona.** Czwarty cel jest bardzo często spotykanym efektem na stronach internetowych. Mówiąc krótko, chcemy, aby było wiadomo, że aktualnie załadowana strona odpowiada stronie, do której prowadzi dana zakładka. Do osiągnięcia tego celu potrzeba trochę więcej pracy — niezbędnych jest pięć stron XHTML. Elementowi `<body>` każdej z nich przypisujemy unikalny identyfikator `ID`. Następnie nadajemy unikalny identyfikator każdemu elementowi `<li>` stanowiącemu odrębną zakładkę. Dzięki tej metodzie będzie można nadać inny styl zakładce odpowiadającej aktualnie załadowanej stronie.

To tyle, jeśli chodzi o teorię. W następnym podrozdziale rozpoczynamy już pracę z kodem. Pamiętaj, że kompletny kod tego projektu dostępny jest na płycie CD dołączonej do książki.

Kod i jego objaśnienie

Aby utworzyć system zakładek za pomocą XHTML i CSS, należy wykonać poniższe polecenia:

1. Tworzymy dokumenty HTML o następującej treści (różnice pomiędzy dokumentami zostały wyróżnione):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns='http://www.w3.org/1999/xhtml' xml:lang='pl'>
    <head>
        <meta http-equiv='Content-Type' content='text/html; charset=iso-8859-2' />
        <title></title>
        <link rel='stylesheet' type='text/css' href='tabs.css' />
        <!-- sztuczka dostosowująca dla przeglądarek  Microsoftu -->
        <!--[if lt IE 7]>
            <script src='/ie7/ie7-standard-p.js' type='text/javascript'></script>
        <![endif]-->
    </head>
    <body id='wrox'>
        <ul id='tabs'>
            <li id='tab1'><a href='wrox.html'><span>Wrox P2P</span></a></li>
            <li id='tab2'><a href='amazon.html'><span>Amazon</span></a></li>
            <li id='tab3'><a href='google.html'><span>Google</span></a></li>
            <li id='tab4'><a href='slashdot.html'><span>Slashdot</span></a></li>
            <li id='tab5'><a href='twit.html'><span>This Week in Tech</span></a></li>
        </ul>
        <div id='iframe'>
            <iframe src='http://p2p.wrox.com' frameborder='0' marginheight='0'
                    marginwidth='0'></iframe>
        </div>
    </body>
</html>
```

2. Zachowujemy pierwszy dokument pod nazwą `wrox.html`.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns='http://www.w3.org/1999/xhtml' xml:lang='pl'>
    <head>
        <meta http-equiv='Content-Type' content='text/html; charset=iso-8859-2' />
        <title></title>
        <link rel='stylesheet' type='text/css' href='tabs.css' />
        <!-- sztuczka dostosowująca dla przeglądarek Microsoftu -->
        <!--[if lt IE 7]>
            <script src='/ie7/ie7-standard-p.js' type='text/javascript'></script>
        <![endif]-->
    </head>
```

```
    <body id='amazon'>
        <ul id='tabs'>
            <li id='tab1'><a href='wrox.html'><span>Wrox P2P</span></a></li>
            <li id='tab2'><a href='amazon.html'><span>Amazon</span></a></li>
            <li id='tab3'><a href='google.html'><span>Google</span></a></li>
            <li id='tab4'><a href='slashdot.html'><span>Slashdot</span></a></li>
            <li id='tab5'><a href='twit.html'><span>This Week in Tech</span></a></li>
        </ul>
        <div id='iframe'>
            <iframe src='http://www.amazon.com'
                        frameborder='0' marginheight='0' marginwidth='0'></iframe>
        </div>
    </body>
</html>
```

3. Drugi dokument zapisujemy pod nazwą `amazon.html`.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns='http://www.w3.org/1999/xhtml' xml:lang='pl'>
    <head>
        <meta http-equiv='Content-Type' content='text/html; charset=iso-8859-2' />
        <title></title>
        <link rel='stylesheet' type='text/css' href='tabs.css' />
        <!-- sztuczka dostosowująca dla przeglądarek Microsoftu -->
        <!--[if lt IE 7]>
            <script src='/ie7/ie7-standard-p.js' type='text/javascript'></script>
        <![endif]-->
    </head>
    <body id='google'>
        <ul id='tabs'>
            <li id='tab1'><a href='wrox.html'><span>Wrox P2P</span></a></li>
            <li id='tab2'><a href='amazon.html'><span>Amazon</span></a></li>
            <li id='tab3'><a href='google.html'><span>Google</span></a></li>
            <li id='tab4'><a href='slashdot.html'><span>Slashdot</span></a></li>
            <li id='tab5'><a href='twit.html'><span>This Week in Tech</span></a></li>
        </ul>
        <div id='iframe'>
            <iframe src='http://www.google.com'
                        frameborder='0' marginheight='0' marginwidth='0'></iframe>
        </div>
    </body>
</html>
```

4. Trzeci dokument zapisujemy jako `google.html`.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns='http://www.w3.org/1999/xhtml' xml:lang='pl'>
    <head>
        <meta http-equiv='Content-Type' content='text/html; charset=iso-8859-2' />
        <title></title>
        <link rel='stylesheet' type='text/css' href='tabs.css' />
        <!-- sztuczka dostosowująca dla przeglądarek Microsoftu -->
        <!--[if lt IE 7]>
            <script src='/ie7/ie7-standard-p.js' type='text/javascript'></script>
        <![endif]-->
```

```
    </head>
    <body id='slashdot'>
        <ul id='tabs'>
            <li id='tab1'><a href='wrox.html'><span>Wrox P2P</span></a></li>
            <li id='tab2'><a href='amazon.html'><span>Amazon</span></a></li>
            <li id='tab3'><a href='google.html'><span>Google</span></a></li>
            <li id='tab4'><a href='slashdot.html'><span>Slashdot</span></a></li>
            <li id='tab5'><a href='twit.html'><span>This Week in Tech</span></a></li>
        </ul>
        <div id='iframe'>
            <iframe src='http://www.slashdot.org'
                        frameborder='0' marginheight='0' marginwidth='0'></iframe>
        </div>
    </body>
</html>
```

5. Czwarty dokument zapiszmy pod nazwą `slashdot.html`.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns='http://www.w3.org/1999/xhtml' xml:lang='pl'>
    <head>
        <meta http-equiv='Content-Type' content='text/html; charset=iso-8859-2' />
        <title></title>
        <link rel='stylesheet' type='text/css' href='tabs.css' />
        <!-- sztuczka dostosowująca dla przeglądarek  Microsoftu -->
        <!--[if lt IE 7]>
            <script src='/ie7/ie7-standard-p.js' type='text/javascript'></script>
        <![endif]-->
    </head>
    <body id='twit'>
        <ul id='tabs'>
            <li id='tab1'><a href='wrox.html'><span>Wrox P2P</span></a></li>
            <li id='tab2'><a href='amazon.html'><span>Amazon</span></a></li>
            <li id='tab3'><a href='google.html'><span>Google</span></a></li>
            <li id='tab4'><a href='slashdot.html'><span>Slashdot</span></a></li>
            <li id='tab5'><a href='twit.html'><span>This Week in Tech</span></a></li>
        </ul>
        <div id='iframe'>
            <iframe src='http://www.twit.tv'
                        frameborder='0' marginheight='0' marginwidth='0'></iframe>
        </div>
    </body>
</html>
```

6. Piąty dokument zapisujemy pod nazwą `twit.html`.

7. Teraz tworzymy arkusz stylów:

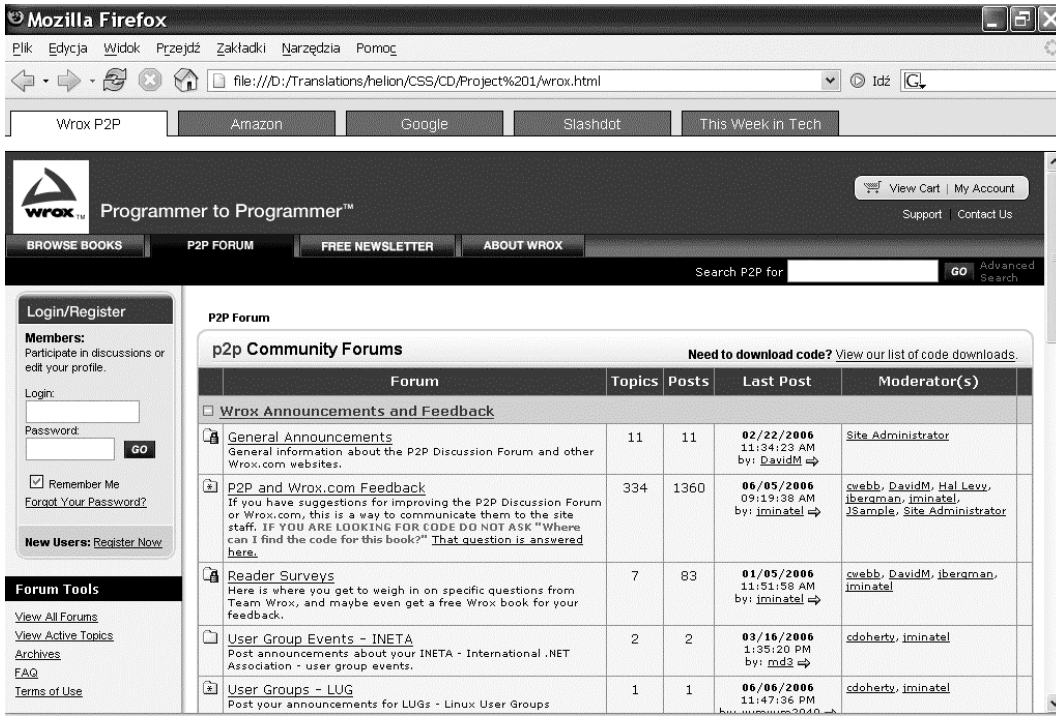
```
body, html {
    margin: 0;
    padding: 0;
}
ul#tabs {
    list-style: none;
    margin: 0;
    padding: 10px 0 0 0;
```

```
    height: 25px;
    border-bottom: 1px solid #000;
    background: #dedede;
}
ul#tabs li {
    float: left;
    margin: 0 5px;
    height: 23px;
    text-align: center;
    position: relative;
    width: 150px;
    border: 1px solid #000;
    top: 1px;
    background: #808080;
}
ul#tabs li:hover {
    border-bottom: 1px solid #fff;
    background: #fff;
}
ul#tabs a {
    display: block;
    height: 100%;
    text-decoration: none;
    color: #fff;
    font: 14px Arial, sans-serif;
}
body#wrox li#tab1,
body#amazon li#tab2,
body#google li#tab3,
body#slashdot li#tab4,
body#twit li#tab5 {
    background: #fff;
    border-bottom: 1px solid #fff;
}
ul#tabs a:hover,
body#wrox li#tab1 a,
body#amazon li#tab2 a,
body#google li#tab3 a,
body#slashdot li#tab4 a,
body#twit li#tab5 a {
    color: #000;
}
ul#tabs span {
    display: block;
    padding: 4px 10px 0 10px;
}
div#iframe {
    position: absolute;
    top: 0;
    bottom: 0;
    right: 0;
    left: 0;
    margin-top: 50px;
    border-top: 1px solid #000;
}
iframe {
    position: absolute;
```

```
    top: 0;
    bottom: 0;
    right: 0;
    left: 0;
    width: 100%;
    height: 100%;
}
```

8. Zapisujemy powyższy arkusz stylów jako `tabs.css`.

Efekt działania powyższego kodu w przeglądarce Mozilla Firefox przedstawiono na rysunku 1.1.



Rysunek 1.1.

Na rysunku 1.1 widać, że zakładka Wrox P2P jest podświetlana, gdy załadowana zostaje strona *wrox.html*. Po kliknięciu innej zakładki, jest ona podświetlana i odpowiednia strona wyświetla się na ekranie, tak jak na rysunku 1.2.

Na rysunku 1.2 widać, że po załadowaniu strony *google.html* podświetliła się zakładka Google.

Przed przejściem do wyjaśniania kodu spójrzmy na poniższy bardzo interesujący fragment:

```
<!-- sztuczka dostosowująca dla przeglądarek Microsoftu -->
<!--[if lt IE 7]>
    <script src='/ie7/ie7-standard-p.js' type='text/javascript'></script>
<![endif]-->
```

Rysunek 1.2.

Ta część kodu pozwala ominąć pewne braki w implementacji CSS w przeglądarkach Internet Explorer 5.5 i 6. Na razie nie będziemy zajmować się bliżej tym kodem (ani testowaniem w przeglądarce Internet Explorer), jako że bardziej szczegółowo jest on omówiony w następnym podrozdziale, zatytułowanym „Testowanie i ograniczenia".

Poniżej znajduje się objaśnienie krok po kroku każdej deklaracji użytej w arkuszu *tabs.css*, co pomoże czytelnikowi dopasować projekt zakładek do własnych potrzeb.

Pierwsza reguła w pliku *tabs.css* służy do usunięcia domyślnych marginesów i dopełnienia z elementu <body>.

```
body, html {
    margin: 0;
    padding: 0;
}
```

Zastosowanie tej reguły nie jest konieczne dla niniejszego projektu, ale została ona dodana dla potrzeb przykładu. Niektóre przeglądarki (IE, Firefox) do elementu <body> domyślnie dodają margines o określonej wartości, inne przeglądarki natomiast (Opera) dodają domyślnie dopełnienie. Ta reguła likwiduje i jedno, i drugie.

Następna reguła służy do nadania odpowiednich właściwości elementowi <ul>, w którym są zawarte wszystkie elementy listy mające służyć jako zakładki:

```
ul#tabs {
    list-style: none;
    margin: 0;
    padding: 10px 0 0 0;
    height: 25px;
    border-bottom: 1px solid #000;
    background: #dedede;
          }
```

Pierwsza deklaracja w powyższej regule to `list-style: none;`. Usuwa ona domyślne formatowanie listy nienumerowanej. Następnie reguła `margin: 0;` usuwa domyślną przestrzeń, jaką każda przeglądarka dodaje wokół elementu `<ul>`. Niestety, tak jak w przypadku elementu `<body>`, różne przeglądarki do elementu `<ul>` dodają albo margines, albo dopełnienie. Deklaracja `padding: 10px 0 0 0;` pozwala na uniknięcie tego problemu, ustawiając 10-pikselowe dopełnienie u góry elementu (przestrzeń pomiędzy górną krawędzią przeglądarki a górną krawędzią każdej zakładki) i usuwając dopełnienie z pozostałych stron. Dodanie tej dodatkowej przestrzeni, tak samo jak w przypadku elementu `<body>`, nie jest konieczne i zależy wyłącznie od indywidualnych potrzeb.

Następna deklaracja nadaje elementowi `<ul>` stałą wysokość. Gdybyśmy tego nie zrobili, to tło i dolna krawędź elementu `<ul>` pojawiłyby się ponad zakładkami zamiast w jednej linii z dolną krawędzią elementów `<li>`. Dzieje się tak, ponieważ elementom `<li>` nadano atrybut `float`, a tego typu elementy nie wpływają w żaden sposób na wysokość elementu nadrzędnego, jako że znajdują się one poza normalnym układem strony i mają wpływ tylko na samą treść elementów, a nie właściwości związane z modelem blokowym takie jak `height`, `padding`, `margin`. Ostatnie dwie deklaracje także nie są niezbędne do implementacji systemu zakładek i można je dowolnie zmieniać. Pierwsza z nich to `border-bottom: 1px solid #000;` i służy ona do ustawienia właściwości dolnej krawędzi wszystkich zakładek. Deklaracja ta potrzebna jest do uzyskania efektu, w którym zakładka odpowiadająca aktualnie załadowanej stronie nie ma krawędzi dolnej, a pozostałe zakładki ją mają. Jak widać na rysunku 1.1, zakładka Wrox P2P jest podświetlona i nie ma krawędzi dolnej. To samo dzieje się z zakładką Google na rysunku 1.2. W uzyskaniu takiego efektu kluczową rolę odgrywa właśnie dolna krawędź elementu `<ul>`. Druga deklaracja to `background: #dedede;`. Tworzy ona kontrast kolorów pomiędzy elementami `<li>` a zawierającym je elementem `<ul>`.

Jeżeli chcemy, aby cały projekt był rzeczywiście przenośny, można zastosować pozycjonowanie absolutne i umieścić go w dowolnym miejscu dokumentu za pomocą odpowiednich atrybutów CSS (`top`, `right`, `bottom`, `left`).

Następna reguła ustawia właściwości wszystkich elementów `<li>`:

```
ul#tabs li {
    float: left;
    margin: 0 5px;
    height: 23px;
    text-align: center;
    position: relative;
    width: 150px;
    border: 1px solid #000;
    top: 1px;
    background: #808080;
}
```

Zakładki w naszym projekcie ułożyliśmy w odpowiedni sposób, posługując się elementami pływającymi na stronie. Dzięki temu wszystkie elementy <li> znalazły się w tej samej linii, jeden obok drugiego, zamiast jeden pod drugim. Efekt ten osiągnęliśmy dzięki zastosowaniu deklaracji `float: left;`. Pozycjonowanie relatywne (`position: relative`) z przesunięciem o jeden piksel od górnej krawędzi zastosowaliśmy, aby dolna krawędź każdej zakładki nachodziła na dolną krawędź zawierającego ją elementu <ul>. Następnie odsunęliśmy od siebie poszczególne zakładki, stosując lewy i prawy margines o szerokości pięciu pikseli oraz zlikwidowaliśmy górny i dolny margines (`margin: 0 5px`). W wyniku tego działania zakładki będą oddalone od siebie o 10 pikseli, a odległość pomiędzy pierwszą zakładką i nadrzędnym elementem <ul> wyniesie pięć pikseli. Następna deklaracja służy do ustawienia stałej wysokości zakładek. Mimo że można się bez tego obejść, to jednak dzięki temu mamy pewność, że zarówno elementy <li>, jak i zawierający je element <ul> mają taką samą wysokość. Zamiast ustawiania stałej wysokości można by było deklarację `top: 1px;` zamienić na `top: 4px;` w celu zapewnienia synchronii pomiędzy dolną krawędzią elementów <li> a dolną krawędzią elementu <ul>. Ustalona wysokość pozwala także na pionowe wycentrowanie tekstu, ale ten sam efekt można osiągnąć poprzez zastosowanie atrybutu `vertical-align` dla każdego elementu <li>. Oczywiście w celu wycentrowania tekstu stosujemy deklarację `text-align: center;`. Następnie za pomocą deklaracji `width: 150px;` ustawiamy stałą szerokość. Zastosowanie tej deklaracji nie jest konieczne, chyba że chcemy, aby każda zakładka miała taką samą szerokość.

> *Jeżeli dla elementu* *zawierającego zakładki zastosowano pozycjonowanie absolutne, to w celu osiągnięcia zgodności z przeglądarką Opera należy temu elementowi nadać szerokość większą niż suma szerokości wszystkich elementów* *w nim zawartych. Konieczność ta w przypadku Opery zachodzi ze względu na fakt, że wszystkie elementy* *są elementami pływającymi, przez co zawierający je element* *nie rozciąga się w poziomie, aby je wszystkie pomieścić. Sytuacja ta spowodowana jest przez pewien dwuznaczny zapis w specyfikacji CSS 2, który powoduje, że przy zastosowaniu wzajemnie wykluczających się interpretacji zarówno Opera, jak i Firefox są zgodne z tą specyfikacją. Konflikt skupia się wokół pozycjonowania absolutnego elementów przy użyciu algorytmu obliczania szerokości oraz tego, w którym momencie należy dokonać jej obliczenia. Opera oblicza szerokość elementów pozycjonowanych absolutnie, zanim weźmie pod uwagę elementy potomne rozmieszczone za pomocą argumentu* `float`*. Firefox natomiast najpierw rozmieszcza elementy przy użyciu* `float`*, a dopiero potem nadaje elementowi pozycjonowanemu absolutnie odpowiednią szerokość.*

Na koniec ustawiamy jeszcze tło i właściwości ramek elementów <li>, aby odróżniały się od otaczającego je elementu <ul>.

Następna reguła powoduje, że element <li> po najechaniu kursorem zmienia kolor tła na biały:

```
ul#tabs li:hover {
    border-bottom: 1px solid #fff;
    background: #fff;
}
```

Powyższa reguła jest bardzo prosta: jeżeli użytkownik najedzie kursorem na którykolwiek z elementów <li>, to tło tego elementu z szarego (ustawionego w poprzedniej regule: `background: #808080;`) zmieni się na białe. Kolor dolnej krawędzi także się zmieni — z czarnego

na biały. Dzięki temu uzyskujemy wrażenie, że podświetlona zakładka pojawia się z przodu, jak na rysunkach 1.1 i 1.2. Jak widać, podświetlona jest tylko zakładka odpowiadająca wczytanej stronie. Gdybyśmy najechali kursorem na pozostałe zakładki, to one też by się podświetliły, ale tylko do czasu usunięcia znad nich kursora.

Następna reguła służy do ustawiania właściwości odnośników wewnątrz elementów <li>:

```
ul#tabs a {
    display: block;
    height: 100%;
    text-decoration: none;
    color: #fff;
    font: 14px Arial, sans-serif;
}
```

Pierwsza deklaracja zmienia sposób formatowania elementów <a> ze śródliniowego (domyślny dla tych elementów) na blokowy (domyślny dla takich elementów jak czy <p>). Dzięki temu odnośniki będą rozciągać się poziomo na całą wolną przestrzeń wewnątrz elementów <li> (od lewej do prawej krawędzi elementu). Następnie deklaracja `height: 100%;` powoduje, że całe wnętrze elementu <li> będzie służyło jako odnośnik. Za pomocą deklaracji `text-decoration: none;` usuwamy domyślne podkreślenie dla odnośników, a deklaracja `color: #fff;` ustawia ich kolor na biały (domyślnie jest zazwyczaj niebieski, a po odwiedzeniu zmienia się na fioletowy). Tak jak i inne deklaracje stosowane w celach czysto kosmetycznych ustawianie koloru odnośników nie jest konieczne. Na koniec zmieniamy właściwości czcionki za pomocą deklaracji `font: 14px Arial, sans-serif;`.

We wszystkich pięciu utworzonych na początku dokumentach elementowi <body> nadaliśmy unikalny identyfikator. Poniższa reguła, przy wykorzystaniu tych identyfikatorów, pozwoli nam na zastosowanie odmiennego stylu dla zakładki odpowiadającej aktualnie wczytanej stronie:

```
body#wrox li#tab1,
body#amazon li#tab2,
body#google li#tab3,
body#slashdot li#tab4,
body#twit li#tab5 {
    background: #fff;
    border-bottom: 1px solid #fff;
}
```

Sama koncepcja jest bardzo prosta. Wykorzystując kaskadowy charakter arkuszy stylów oraz unikalne nazewnictwo każdej strony, jesteśmy w stanie nadać odmienny styl zakładce odpowiadającej aktualnie wczytanej stronie. Same zakładki muszą także posiadać niepowtarzalne nazwy. W połączeniu z identyfikatorem elementu <body> reguła `body#wrox li#tab1` nadpisuje poprzednią regułę (`ul#tabs li`), ponieważ posiada ona większą precyzję zapisu. Jeżeli element <body> będzie miał identyfikator `wrox`, a element <li> będzie miał identyfikator `tab1`, to nastąpi nadpisanie reguły `ul#tabs li` przez regułę `body#wrox li#tab1`, co z kolei prowadzi do zastosowania białego tła i jednolitej białej krawędzi dolnej dla zakładki odpowiadającej aktualnie załadowanej stronie.

Przedstawiona powyżej reguła jest identyczna z regułą `ul#tabs li:hover`, o której była już mowa wcześniej. Mimo że jest to znakomita okazja do zastosowania grupowania selektorów,

to napisaliśmy je oddzielnie, aby oddzielić od siebie dwa różne zagadnienia. Przykład grupowania widzimy w następnej regule:

```
ul#tabs a:hover,
body#wrox li#tab1 a,
body#amazon li#tab2 a,
body#google li#tab3 a,
body#slashdot li#tab4 a,
body#twit li#tab5 a {
    color: #000;
}
```

Powyższa reguła rozpoczyna się listą sześciu selektorów zgrupowanych w celu uniknięcia powtarzania. Pierwszy selektor dopasowuje się do wszystkich elementów <a>, które są potomkami elementu <ul> o identyfikatorze tabs, ale tylko w momencie, gdy kursor myszy znajduje się nad elementem <a>. Pozostałe selektory, jak już widzieliśmy w poprzedniej regule, dopasowują się do odnośników aktualnie załadowanej strony w oparciu o identyfikator elementu <body> i identyfikator elementu <li>. Ze względu na fakt, że kolor odnośników w elementach <li> został ustawiony na biały, i tło każdego z tych elementów zmienia się na białe po najechaniu na zakładkę kursorem lub kiedy dana zakładka odpowiada aktualnie załadowanej stronie. W tych sytuacjach kolor odnośnika musi się zmieniać, aby był on widoczny na białym tle. Jeżeli wszystkie warunki zostaną spełnione, stosowana jest deklaracja color: #000;.

Następna reguła służy do nadania odpowiedniego formatowania elementowi zagnieżdżonemu wewnątrz każdego elementu <a>:

```
ul#tabs span {
    display: block;
    padding: 4px 10px 0 10px;
}
```

Pierwsza deklaracja w powyższej regule zmienia typ elementu z domyślnego śródliniowego na blokowy. Dzięki temu można zastosować dopełnienie odpowiadające modelowi blokowemu zamiast dopełnienia odpowiadającego modelowi śródliniowemu. Robimy to, aby móc kontrolować wolną przestrzeń wokół tekstu na każdej zakładce. Czynności te mogą, ale nie muszą być potrzebne, w zależności od indywidualnych potrzeb projektu.

Możliwe, że się zastanawiasz, czemu zamiast elementu nie użyliśmy elementu . Otóż z semantycznego punktu widzenia niedozwolone jest umieszczanie elementów blokowych, takich jak np. , wewnątrz elementów śródliniowych, takich jak <a> (mimo że element <a> został przez nas zmieniony na element blokowy). Różne przeglądarki mogłyby się różnie zachować w takiej sytuacji. Z punktu widzenia zgodności ze standardami nie ma żadnych przeciwwskazań, jeśli chodzi o umieszczanie elementów wewnątrz elementów <a>, ponieważ elementy śródliniowe można bez problemu wzajemnie zagnieżdżać. W tym projekcie bierzemy pod uwagę błędy zgłaszane przez walidator W3C znajdujący się pod adresem *http://validator.w3.org*. Program ten używany jest przez twórców stron internetowych w celu sprawdzania, czy tworzone przez nich dokumenty zgodne są z obowiązującymi standardami.

Element nie jest konieczny do tego, aby projekt mógł poprawnie funkcjonować. Tak jak już wspomniana wcześniej stała wysokość zakładek, tak też użycie elementu do

kontrolowania pionowego położenia tekstu (w odróżnieniu od położenia poziomego) na zakładkach jest opcjonalne.

Pozostałe dwie reguły nie mają nic wspólnego z samym systemem zakładek i zostały dodane wyłącznie w celu wypełnienia pustej części dokumentu treścią poprzez wczytanie dokumentów zewnętrznych z popularnej strony za pomocą ramki wewnętrznej.

Testowanie i ograniczenia

W niniejszym podrozdziale opisuję problemy, które mogą wystąpić przy testowaniu projektu w różnych przeglądarkach. Zazwyczaj podczas pracy nad projektem ograniczam się do testowania go tylko w przeglądarce Firefox, dopóki nie nabierze on kształtów i nie zacznie zbliżać się do końca. Dochodząc do tego momentu, rozszerzam zestaw przeglądarek testowych, dodając Internet Explorera, Operę, Safari itd. W przypadku przeglądarek dobrze obsługujących standardy (Opera, Safari) zazwyczaj potrzebne są tylko niewielkie ulepszenia. Problem rozpoczyna się, gdy dochodzi do testowania pod IE, która albo bardzo słabo obsługuje niektóre standardy, albo wcale ich nie obsługuje, dlatego też w niniejszym rozdziale skupiam się przede wszystkim na tej aplikacji.

W poprzednim podrozdziale wspominaliśmy niewielki fragment kodu, którego zadaniem było pomóc nam sprawić, aby przeglądarka Internet Explorer w wersjach 5.5 i 6 była kompatybilna z niektórymi nieobsługiwanymi przez nią właściwościami CSS. Poniżej przypominam omawiany fragment kodu:

```
<!-- sztuczka dostosowująca dla przeglądarek  Microsoftu -->
<!--[if lt IE 7]>
     <script src='/ie7/ie7-standard-p.js' type='text/javascript'></script>
<![endif]-->
```

Osoby, które czytały moją książkę *Beginning CSS: Cascading Style Sheets for Web Design* (wydaną przez wydawnictwo Wiley Publishing) prawdopodobnie pamiętają, że w rozdziale 18. pisałem, jak pobrać i zainstalować bibliotekę JavaScript IE7. Jako że ta książka przeznaczona jest dla bardziej zawansowanych czytelników, nie będę się tu szczegółowo rozpisywał na temat instalacji tej biblioteki, a ograniczę się tylko do ogólnego przedstawienia procesu instalacji oraz pobieżnego przedstawienia jej możliwości.

Biblioteka JavaScript IE7 została napisana przez londyńczyka, Deana Edwardsa, w celu ułatwienia twórcom stron internetowych radzenia sobie ze słabą obsługą CSS przez przeglądarkę Internet Explorer, która już od pięciu lat nie doczekała się żadnych poważniejszych uaktualnień w tym kierunku. Edwards przy użyciu JavaScript dokonuje implementacji właściwości CSS oryginalnie nieobsługiwanych przez przeglądarkę Internet Explorer w wersjach 5.5 i 6. Dzięki temu IE zbliża się funkcjonalnością do innych, jak na razie o wiele lepiej obsługujących standardy, przeglądarek, takich jak Safari czy Firefox. Biblioteka IE7 napisana jest w sposób bardzo przejrzysty, dzięki czemu wydaje się, że Internet Explorer rzeczywiście obsługuje te właściwości CSS, które do tej pory sprawiały mu problemy. Ponadto osoby korzystające z tej biblioteki nie muszą znać JavaScript w stopniu większym niż potrzeba, aby ją włączyć. Niektóre z najważniejszych właściwości CSS, których obsługę przez IE umożliwia lub naprawia biblioteka IE7, to:

- atrybuty `min-width`, `max-width` oraz `min-height`,
- pseudoklasy `:hover`, `:active` oraz `:focus` (działają nie tylko z elementem `<a>`),
- różne zaawansowane selektory, takie jak bezpośredni selektor potomka (`>`), selektory atrybutów (`input[type]`), selektor przylegających elementów równorzędnych (`+`) oraz selektor równorzędnych elementów przylegających pośrednio (`-`),
- pseudoklasy strukturalne, takie jak: `:root`, `:first-child` oraz `:last-child`,
- pseudoelementy `::before` i `::after` oraz atrybut `content`.

Oczywiście to nie wszystkie możliwości klasy IE7. Powyżej podałem tylko kilka przykładów. To, co jest najbardziej zaskakujące, to rozmiar i szybkość działania tej modularnej budowy biblioteki. Dzięki takiej konstrukcji twórcy uaktywniają tylko potrzebne funkcje, co znacznie zmniejsza ilość koniecznych do pobrania danych. Biblioteka główna zajmuje tylko około 24 kB. Naprawia lub dodaje obsługę podanych wyżej właściwości (oprócz `:last-child`, która znajduje się w specjalnej bibliotece selektorów CSS3).

Bibliotekę JavaScript IE7 można pobrać ze strony organizacji SourceForge, która na swoich serwerach udostępnia tysiące projektów typu open source. Dokładny adres do pobrania biblioteki IE7 JavaScript to: *https://sourceforge.net/project/showfiles.php?group_id=109983*.

Po ukończeniu pobierania biblioteki IE7 należy ją rozpakować i umieścić w katalogu głównym serwera. Na przykład, jeżeli strona znajduje się pod adresem *http://www.example.com/*, to biblioteka IE7 powinna znajdować się w katalogu dostępnym pod adresem *http://www.example.com/IE7/*. Klasę tę można umieścić także w innym katalogu, ale wtedy należy pamiętać o zmianie przykładów kodu w tej książce tak, aby odpowiadały one ścieżce do katalogu z biblioteką.

Biblioteka JavaScript IE7 jest projektem typu open source i jest dostępna na zasadach licencji Creative Commons LGPL (ang. *Lesser General Public Licence*). Pełny tekst tej licencji można przeczytać na stronie *http://creativecommons.org/licenses/LGPL/2.1*.

Więcej informacji na temat biblioteki JavaScript IE7 można znaleźć na specjalnie jej poświęconej stronie pod adresem: *http://dean.edwards.name/ie7*.

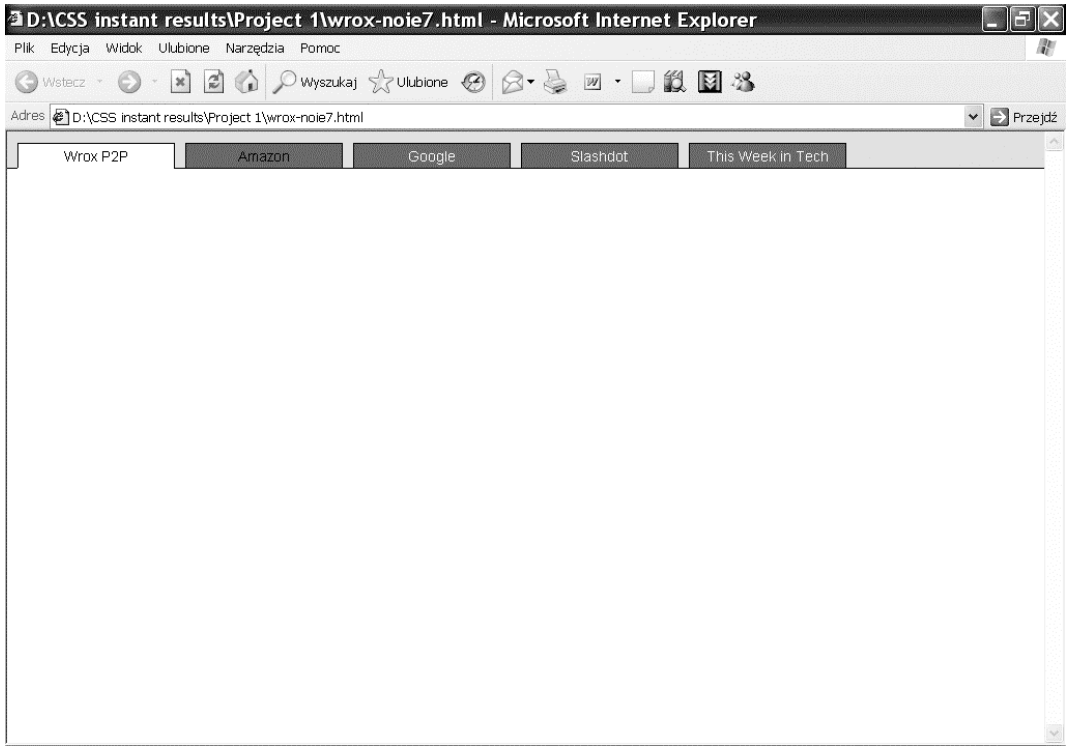
Następny podrozdział poświęcony jest analizie przydatnej w naszym projekcie funkcjonalności biblioteki Edwardsa.

Co daje nam biblioteka IE7?

Aby się przekonać, co daje nam biblioteka IE7, wystarczy otworzyć nasz projekt w przeglądarce Internet Explorer bez instalowania tej biblioteki. Rezultat otwarcia strony *wrox.html* bez funkcjonalności IE7 w Internet Explorerze 6 pokazano na rysunku 1.3.

Różnice nie są wielkie poza dwoma wyjątkami:

- kolor tła zakładek nie zmienia się po najechaniu na nie kursorem,
- zawartość ramki `<iframe>` nie jest widoczna.



Rysunek 1.3.

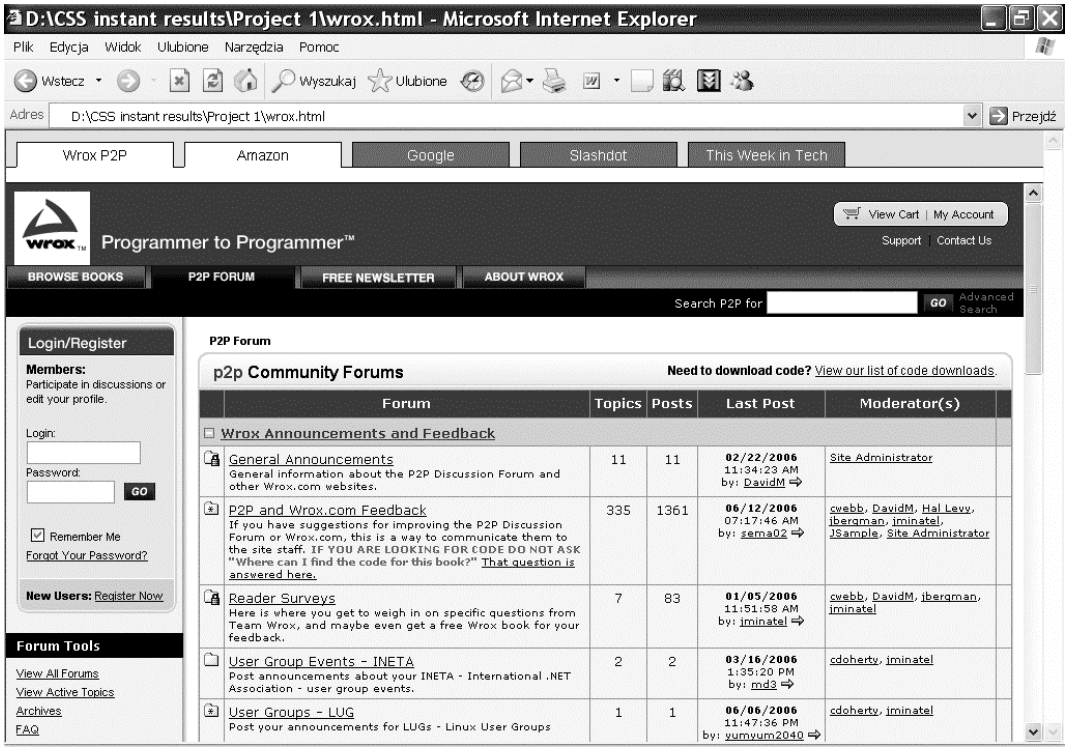
Pierwszy problem wystąpił dlatego, że Internet Explorer obsługuje pseudoklasę `:hover` tylko dla elementów `<a>`, a nie tak jak Firefox, Opera i Safari dla wszystkich elementów. Drugi problem ma związek z tym, że Internet Explorer nie radzi sobie z kombinacją atrybutów `top`, `right`, `bottom` oraz `left` wpływających na rozmiary elementów pozycjonowanych absolutnie lub elementów o ustalonym położeniu. Dzięki zastosowaniu biblioteki IE7 Internet Explorer nie ustępuje funkcjonalnością innym przeglądarkom (Opera, Firefox, Safari). Na rysunku 1.4 pokazano tę samą stronę w Internet Explorerze 6, ale już po zainstalowaniu biblioteki IE7.

Na rysunku 1.4 widać, że tło zakładek zmienia kolor po najechaniu na nie kursorem, a także, że wyświetla się zawartość elementu `<iframe>`.

W następnym podrozdziale pokażę różne sposoby modyfikacji projektu.

Używanie i modyfikacja projektu

W niniejszym podrozdziale prezentuję alternatywne podejścia do projektu zakładek, gdyż nie każdemu może on się podobać w obecnej postaci. Alternatywne podejścia analizowane w niniejszym podrozdziale są następujące:



Rysunek 1.4.

- Wykorzystanie obrazków w tle zamiast kolorów i prostokątnych obramowań.
- Wykorzystanie obrazków w tle, które zawierają tekst, bez utraty dostępności.
- Takie użycie obrazów w tle, aby zakładki były elastyczne i mogły zmieniać rozmiary w określonych granicach.

Oczywiście te alternatywne podejścia w pewnym stopniu komplikują nasz projekt, ale ich celem jest dostarczenie jak największej elastyczności oraz pokazanie możliwie największej liczby implementacji zakładek.

Pierwsza modyfikacja będzie dotyczyła zastosowania obrazków w tle zamiast zwykłych kolorów i prostokątnych obramowań.

Zakładki z obrazkami w tle

Aby utworzyć zakładki z obrazkami w tle, należy wykonać następujące kroki (kod źródłowy do tego projektu znajduje się na dołączonej do książki płycie CD w podkatalogu *with-background* w katalogu *Project 1*):

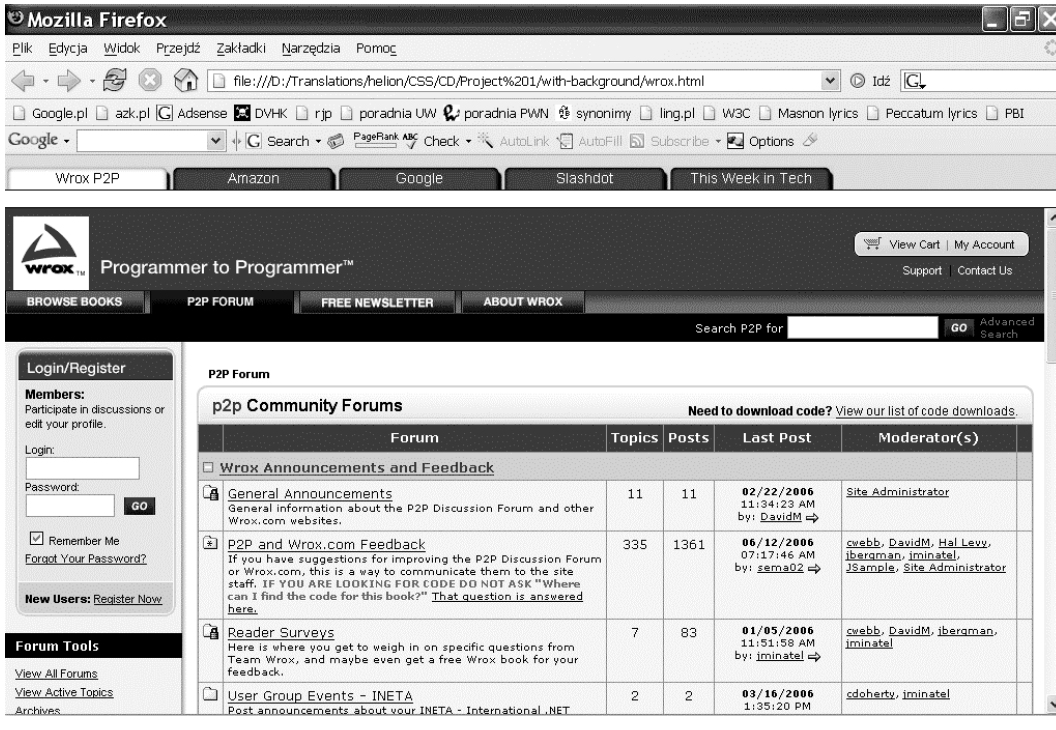
1. Tworzymy następujący arkusz stylów (zmiany w stosunku do jego poprzedniej wersji zostały wyróżnione):

```
body, html {
    margin: 0;
    padding: 0;
}
ul#tabs {
    list-style: none;
    margin: 0;
    padding: 10px 0 0 0;
    height: 23px;
    border-bottom: 1px solid #000;
    background: #dadada;
}
ul#tabs li {
    float: left;
    margin: 0;
    height: 23px;
    text-align: center;
    width: 160px;
    background: transparent url('images/tab.png') no-repeat scroll top;
}
ul#tabs a {
    display: block;
    height: 100%;
    text-decoration: none;
    color: #fff;
    font: 14px Arial, sans-serif;
}
ul#tabs li:hover,
body#wrox li#tab1,
body#amazon li#tab2,
body#google li#tab3,
body#slashdot li#tab4,
body#twit li#tab5 {
    background: transparent url('images/tab_hover.png') no-repeat scroll top;
}
ul#tabs a:hover,
body#wrox li#tab1 a,
body#amazon li#tab2 a,
body#google li#tab3 a,
body#slashdot li#tab4 a,
body#twit li#tab5 a {
    color: #000;
}
ul#tabs span {
    display: block;
    padding: 4px 10px 0 10px;
}
div#iframe {
    position: absolute;
    top: 0;
    bottom: 0;
    right: 0;
    left: 0;
    margin-top: 50px;
    border-top: 1px solid #000;
}
```

```
iframe {
    position: absolute;
    top: 0;
    bottom: 0;
    right: 0;
    left: 0;
    width: 100%;
    height: 100%;
}
```

2. Zapisujemy powyższy arkusz pod nazwą `tabs.css`.

Rezultat tej modyfikacji widoczny jest na rysunku 1.5.



Rysunek 1.5.

Liczba modyfikacji potrzebnych do ustawienia obrazków w tle zamiast kolorów i obramowań jest zaskakująco mała. Poniżej znajduje się wyjaśnienie krok po kroku każdej z dokonanych zmian.

Pierwsza zmiana dotyczy wysokości elementu `<ul>`, która równa jest wysokości obrazka użytego jako tło elementów `<li>`:

```
ul#tabs {
    list-style: none;
    margin: 0;
    padding: 10px 0 0 0;
    height: 23px;
```

```
    border-bottom: 1px solid #000;
    background: #dadada;
}
```

Druga modyfikacja dotyczy koloru tła elementu `<ul>` zawierającego wszystkie pięć zakładek. Tło jest nieznacznie jaśniejsze, aby pasowało do utworzonych w programie Adobe Photoshop obrazków tła. Mimo zastosowania identycznej wartości `#dadada` zarówno w arkuszu stylów, jak i w programie Adobe Photoshop, przy tworzeniu obrazków tła, kolory nie pasowały do siebie w żadnej z przeglądarek. Aby zniwelować tę różnicę, po prostu nieznacznie przyciemniliśmy szare tło elementu `<ul>`.

Kolejna modyfikacja dotyczy elementów `<li>`:

```
ul#tabs li {
    float: left;
    margin: 0;
    height: 23px;
    text-align: center;
    width: 160px;
    background: transparent url('images/tab.png') no-repeat scroll top;
}
```

Najpierw usuwamy margines spomiędzy elementów `<li>` oraz niepotrzebne nam już deklaracje: `position: relative;` oraz `top: 1px;`. Poprzednio prawy i lewy margines elementów `<li>` wynosił 5 pikseli. Obecnie nie potrzebujemy już tego marginesu, bo możemy go dodać za pomocą obrazków w tle. Po marginesach przyszła kolej na zmianę szerokości na 160 pikseli ze względu na fakt usunięcia lewego i prawego marginesu elementów `<li>`. Ostatnia deklaracja służy do ustawienia obrazka jako tła zakładek. Rozmiar obrazka odpowiada rozmiarowi elementu `<li>`, czyli wynosi 160×23 piksele. Zmieniając wymiary zakładek, należy pamiętać o atrybutach `width` i `height` pojawiających się w powyższej regule, a także o atrybucie `height` zdefiniowanym w regule `ul#tabs` (wszelkie modyfikacje rozmiarów zakładek należy rozpoczynać od modyfikacji tego atrybutu). Dodatkowo, zmieniając rozmiar zakładek, należy wziąć pod uwagę wartości dopełnienia zastosowanego dla elementu `<span>`, w obrębie którego znajduje się tekst wyświetlany na zakładkach, a który służy do kontrolowania wyrównania tekstu w pionie.

Ostatnie zmiany dotyczą zakładki odpowiadającej aktualnie załadowanej stronie lub obrazka, który pojawia się w tle po najechaniu na zakładkę kursorem:

```
ul#tabs li:hover,
body#wrox li#tab1,
body#amazon li#tab2,
body#google li#tab3,
body#slashdot li#tab4,
body#twit li#tab5 {
    background: transparent url('images/tab_hover.png') no-repeat scroll top;
}
```

W pierwszym arkuszu stylów reguła `ul#tabs li:hover` występowała oddzielnie. Teraz zgrupowaliśmy ją z regułą definiującą zakładkę odpowiadającą aktualnie załadowanej stronie oraz zmodyfikowaliśmy deklarację tła w taki sposób, aby wskazywała na obrazek `tab_hover.png`.

I to by było na tyle. W następnym punkcie umieścimy w tle obrazki zawierające tekst, nie wpływając przy tym ujemnie na dostępność.

Obrazki w tle zawierające tekst

W celu utworzenia zakładek z obrazkami w tle zawierającymi tekst należy wykonać następujące kroki (kod źródłowy do tego projektu znajduje się na dołączonej do książki płycie CD w podkatalogu *with-text* w katalogu *Project 1*).

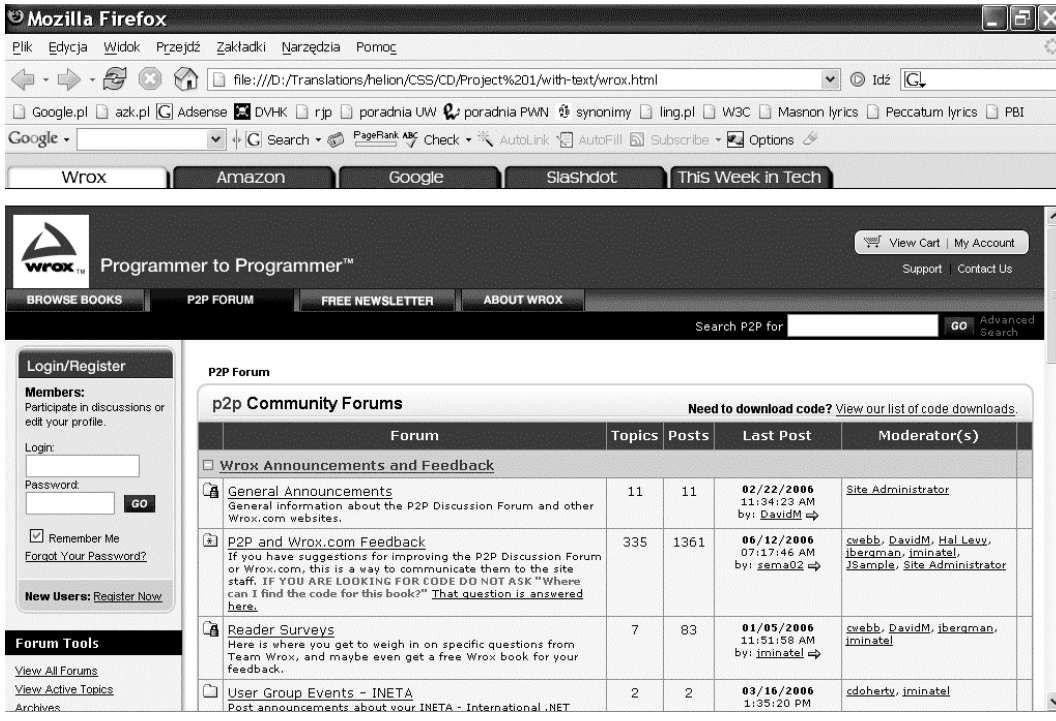
1. Wprowadź następujące zmiany w pliku *tabs.css*. Zmiany w stosunku do poprzedniego arkusza (używanego do ustawiania obrazków w tle) zostały wyróżnione. Niektóre deklaracje usunięto całkowicie:

```
body, html {
    margin: 0;
    padding: 0;
}
ul#tabs {
    list-style: none;
    margin: 0;
    padding: 10px 0 0 0;
    height: 23px;
    border-bottom: 1px solid #000;
    background: #dadada;
}
ul#tabs li {
    float: left;
    margin: 0;
    height: 23px;
    text-align: center;
    width: 160px;
}
ul#tabs a {
    display: block;
    height: 100%;
    text-decoration: none;
    color: #fff;
    font: 14px Arial, sans-serif;
}
ul#tabs li#tab1 {
    background: transparent url('images/wrox-tab.png') no-repeat scroll top;
}
ul#tabs li#tab1:hover,
body#wrox li#tab1 {
    background: transparent url('images/wrox-tab-hover.png') no-repeat scroll top;
}
ul#tabs li#tab2 {
    background: transparent url('images/amazon-tab.png') no-repeat scroll top;
}
ul#tabs li#tab2:hover,
body#amazon li#tab2 {
    background: transparent url('images/amazon-tab-hover.png') no-repeat scroll top;
}
```

```
ul#tabs li#tab3 {
    background: transparent url('images/google-tab.png') no-repeat scroll top;
}
ul#tabs li#tab3:hover,
body#google li#tab3 {
    background: transparent url('images/google-tab-hover.png') no-repeat scroll top;
}
ul#tabs li#tab4 {
    background: transparent url('images/slashdot-tab.png') no-repeat scroll top;
}
ul#tabs li#tab4:hover,
body#slashdot li#tab4 {
    background: transparent url('images/slashdot-tab-hover.png') no-repeat scroll
top;
}
ul#tabs li#tab5 {
    background: transparent url('images/twit-tab.png') no-repeat scroll top;
}
ul#tabs li#tab5:hover,
body#twit li#tab5 {
    background: transparent url('images/twit-tab-hover.png') no-repeat scroll top;
}
ul#tabs a:hover,
body#wrox li#tab1 a,
body#amazon li#tab2 a,
body#google li#tab3 a,
body#slashdot li#tab4 a,
body#twit li#tab5 a {
    color: #000;
}
ul#tabs span {
    display: block;
    padding: 4px 10px 0 10px;
    visibility: hidden;
}
div#iframe {
    position: absolute;
    top: 0;
    bottom: 0;
    right: 0;
    left: 0;
    margin-top: 50px;
    border-top: 1px solid #000;
}
iframe {
    position: absolute;
    top: 0;
    bottom: 0;
    right: 0;
    left: 0;
    width: 100%;
    height: 100%;
}
```

2. Zapisujemy powyższy arkusz jako `tabs.css`.

Rezultat zastosowania tych zmian przedstawiono na rysunku 1.6.



Rysunek 1.6.

Mimo że zmiany dokonane w tym przypadku są dużo większe niż poprzednio, kiedy w tle zakładek ustawialiśmy obrazek bez tekstu, to większość spośród nich dotyczy ustawiania innego obrazka w tle każdej zakładki.

```
ul#tabs li#tab1 {
    background: transparent url('images/wrox-tab.png') no-repeat scroll top;
}
ul#tabs li#tab1:hover,
body#wrox li#tab1 {
    background: transparent url('images/wrox-tab-hover.png') no-repeat scroll top;
}
```

Powyższy fragment kodu służy do ustawiania właściwości tła pierwszej zakładki, która jest odnośnikiem do strony Wrox P2P. Najpierw usuwamy deklarację `background` z reguły `ul#tabs li`, następnie, podobnie jak dla pierwszej zakładki, definiujemy reguły dla wszystkich pozostałych zakładek, ponieważ wszystkie zakładki mają teraz tekst umieszczony na obrazku znajdującym się w ich tle.

W dalszej części arkusza stylów, w celu zachowania dostępności, ukrywamy tekst pojawiający się na zakładkach. Nie usuwając tekstu, tylko go ukrywając, powodujemy, że nasza strona nie traci na dostępności.

```
ul#tabs span {
    display: block;
    padding: 4px 10px 0 10px;
    visibility: hidden;
}
```

Takie ukrywanie tekstu powoduje, że jest on niewidoczny dla ludzkiego oka, ale widoczny dla robotów wyszukujących (np. Googlebota) i przeglądarek dla niewidomych (np. JAWS), które jeszcze nie obsługują CSS. Dzięki temu nasza strona jest tak samo dostępna jak wcześniej.

Ostatnia modyfikacja dotyczy tworzenia elastycznych zakładek, które rozciągają się i kurczą w zależności od potrzeby.

Zakładki elastyczne

Aby stworzyć zakładki elastyczne, które się rozciągają, wypełniając całą dostępną im przestrzeń, należy wykonać następujące czynności:

1. Otwieramy plik *tabs.css* i dokonujemy w nim następujących zmian (zmiany w stosunku do poprzedniego projektu zostały wyróżnione):

```
body, html {
    margin: 0;
    padding: 0;
}
ul#tabs {
    list-style: none;
    margin: 0;
    padding: 10px 0 0 0;
    height: 23px;
    border-bottom: 1px solid #000;
    background: #dadada;
    position: relative;
}
ul#tabs li {
    position: absolute;
    bottom: 0;
    margin: 0;
    height: 23px;
    text-align: center;
    width: 20%;
    background: transparent url('images/tab/tab_01.png') no-repeat scroll left;
}
ul#tabs li > div {
    height: 23px;
    background: transparent url('images/tab/tab_03.png') no-repeat scroll right;
}
ul#tabs li > div > div {
    height: 23px;
    background: transparent url('images/tab/tab_02.png') repeat-x scroll center;
    margin: 0 8px 0 7px;
}
```

```
ul#tabs a {
    display: block;
    height: 100%;
    text-decoration: none;
    color: #fff;
    font: 14px Arial, sans-serif;
}
ul#tabs li#tab1 {
    left: 0;
}
ul#tabs li#tab2 {
    left: 20%;
}
ul#tabs li#tab3 {
    left: 40%;
}
ul#tabs li#tab4 {
    left: 60%;
}
ul#tabs li#tab5 {
    left: 80%;
}
ul#tabs li:hover,
body#wrox li#tab1,
body#amazon li#tab2,
body#google li#tab3,
body#slashdot li#tab4,
body#twit li#tab5 {
    background: transparent url('images/tab-hover/tab-hover_01.png') no-repeat scroll
left;
}
ul#tabs li:hover > div,
body#wrox li#tab1 > div,
body#amazon li#tab2 > div,
body#google li#tab3 > div,
body#slashdot li#tab4 > div,
body#twit li#tab5 > div {
    background: transparent url('images/tab-hover/tab-hover_03.png') no-repeat scroll
right;
}
ul#tabs li:hover > div > div,
body#wrox li#tab1 > div > div,
body#amazon li#tab2 > div > div,
body#google li#tab3 > div > div,
body#slashdot li#tab4 > div > div,
body#twit li#tab5 > div > div {
    background: transparent url('images/tab-hover/tab-hover_02.png') repeat-x scroll
center;
}
ul#tabs a:hover,
body#wrox li#tab1 a,
body#amazon li#tab2 a,
body#google li#tab3 a,
body#slashdot li#tab4 a,
body#twit li#tab5 a {
    color: #000;
}
```

```
ul#tabs span {
    display: block;
    padding: 4px 10px 0 10px;
}
div#iframe {
    position: absolute;
    top: 0;
    bottom: 0;
    right: 0;
    left: 0;
    margin-top: 50px;
    border-top: 1px solid #000;
}
iframe {
    position: absolute;
    top: 0;
    bottom: 0;
    right: 0;
    left: 0;
    width: 100%;
    height: 100%;
}
```

2. Zapisujemy plik *tabs.css*.

3. Następnie otwieramy pliki *wrox.html*, *amazon.html*, *google.html*, *slashdot.html* oraz *twit.html* i wprowadzamy w nich następujące zmiany:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns='http://www.w3.org/1999/xhtml' xml:lang='pl'>
    <head>
        <meta http-equiv='Content-Type' content='text/html; charset=iso-8859-2' />
        <title></title>
        <link rel='stylesheet' type='text/css' href='tabs.css' />
        <!--sztuczka dostosowująca dla przeglądarek Microsoftu -->
        <!--[if lt IE 7]>
            <link rel='stylesheet' type='text/css' href='tabs-ie.css' />
            <script src='/ie7/ie7-standard-p.js' type='text/javascript'></script>
        <![endif]-->
    </head>
    <body id='wrox'>
        <ul id='tabs'>
            <li id='tab1'>
                <div><div>
                    <a href='wrox.html'><span>Wrox P2P</span></a>
                </div></div>
            </li>
            <li id='tab2'>
                <div><div>
                    <a href='amazon.html'><span>Amazon</span></a>
                </div></div>
            </li>
            <li id='tab3'>
                <div><div>
                    <a href='google.html'><span>Google</span></a>
```

```
                </div></div>
            </li>
            <li id='tab4'>
                <div><div>
                    <a href='slashdot.html'><span>Slashdot</span></a>
                </div></div>
            </li>
            <li id='tab5'>
                <div><div>
                    <a href='twit.html'><span>This Week in Tech</span></a>
                </div></div>
            </li>
        </ul>
        <div id='iframe'>
            <iframe src='http://p2p.wrox.com'
                        frameborder='0' marginheight='0' marginwidth='0'></iframe>
        </div>
    </body>
</html>
```

4. Zapisujemy pliki *wrox.html*, *amazon.html*, *google.html*, *slashdot.html* oraz *twit.html*.

5. Tworzymy nowy arkusz stylów i wpisujemy następującą regułę:

```
ul#tabs {
    height: 22px;
}
```

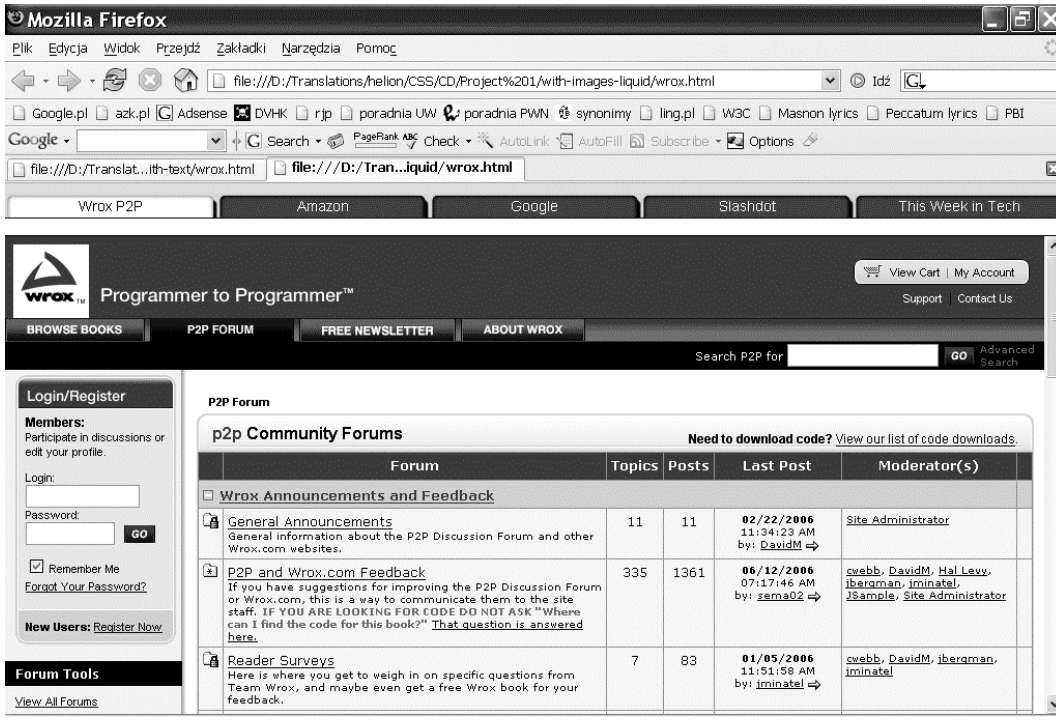
6. Zapisujemy powyższy arkusz stylów jako `tabs-ie.css`.

Rezultat tego kodu przedstawiono na rysunku 1.7.

Ostatnia modyfikacja projektu jest najbardziej skomplikowana, ale zarazem daje pewną elastyczność, ponieważ można ją łatwo zastosować w projektach, które wymagają dopasowania wielkości zakładek do rozmiaru elementów, w których się one znajdują. W tym przykładzie nie ma ograniczeń co do minimalnej lub maksymalnej wielkości zakładki. Takie ograniczenia moglibyśmy stworzyć, stosując właściwości `min-width` oraz `max-width` w regule `ul#tabs`.

Pierwsza modyfikacja dokonana w pliku *tabs.css* polegała na dodaniu deklaracji `position: relative;` do reguły `ul#tabs`.

```
ul#tabs {
    list-style: none;
    margin: 0;
    padding: 10px 0 0 0;
    height: 23px;
    border-bottom: 1px solid #000;
    background: #dadada;
    position: relative;
}
```



Rysunek 1.7.

Dzięki dodaniu deklaracji `position: relative;` elementy `<li>` pozycjonowane absolutnie, znajdujące się wewnątrz elementu `<ul>` o identyfikatorze `tabs`, będą pozycjonowane relatywnie względem tego elementu. Zastosowałem tutaj pozycjonowanie ze względu na fakt, że Internet Explorer błędnie obsługuje wartości procentowe. Mimo ustawienia szerokości każdego elementu `<li>` na 20%, ostatni z elementów w Internet Explorerze pojawia się w nowej linii — a przecież 20%*5 = 100%!

```
ul#tabs li {
    position: absolute;
    bottom: 0;
    margin: 0;
    height: 23px;
    text-align: center;
    width: 20%;
    background: transparent url('images/tab/tab_01.png') no-repeat scroll left;
}
```

Pierwsza deklaracja w powyższej regule powoduje, że elementy `<li>` będą pozycjonowane absolutnie względem nadrzędnego elementu `<ul>`. Deklaracja `bottom: 0;` ustawia każdy element `<li>` względem dolnej krawędzi zawierającego je elementu `<ul>`. Następna deklaracja (`width: 20%;`) ustawia szerokość każdego elementu `<li>` na 20% szerokości otaczającego elementu `<ul>`. Problem z Internet Explorerem udało się ominąć dzięki absolutnemu pozycjonowaniu elementów `<li>`, przez co ostatni z nich nie może zostać przeniesiony do nowej linii.

Ostatnia deklaracja służy do ustawienia w tle pierwszego obrazka. Przy użyciu programu Photoshop pociąłem obrazek na trzy części. Pierwsza będzie służyła jako lewa strona i lewy róg zakładki, druga jako środkowa część zakładki, a trzecia jako prawa strona i prawy róg. Pierwszy obrazek zostaje ustawiony (i wyrównany do lewej) jako niepowtarzające się tło wszystkich elementów <li>. Następnie w celu dodania możliwości rozciągania się zakładek zastosowaliśmy dodatkowe elementy w każdym z dokumentów XHTML. Elementy te posłużą nam jako nośniki utworzonych wcześniej obrazków.

```
ul#tabs li > div {
    height: 23px;
    background: transparent url('images/tab/tab_03.png') no-repeat scroll right;
}
```

Następna reguła służy do ustawienia prawej strony zakładki. Do tego posłuży nam prawa strona pierwszego elementu , któremu nadaliśmy wysokość 23 pikseli, czyli taką samą jak wysokość zawierającego go elementu <li>. Następnie ustawiamy obrazek w tle pierwszego elementu , wyrównując go do prawej. Drugi element jest zagnieżdżony w pierwszym i służy do ustawienia środkowej części zakładki:

```
ul#tabs li > div > div {
    height: 23px;
    background: transparent url('images/tab/tab_02.png') repeat-x scroll center;
    margin: 0 8px 0 7px;
}
```

Temu elementowi również nadajemy wysokość 23 pikseli. Tło zostało wyrównane do środka elementu i będzie się powtarzało wzdłuż osi *x*, czyli poziomo. Na koniec, w celu uniknięcia nachodzenia na siebie elementów, stosujemy lewy i prawy margines o szerokości odpowiadającej szerokości obrazków użytych po lewej i prawej stronie zakładki.

Następne pięć reguł służy do pozycjonowania w poziomie wszystkich elementów <li>:

```
ul#tabs li#tab1 {
    left: 0;
}
ul#tabs li#tab2 {
    left: 20%;
}
ul#tabs li#tab3 {
    left: 40%;
}
ul#tabs li#tab4 {
    left: 60%;
}
ul#tabs li#tab5 {
    left: 80%;
}
```

Każda zakładka jest wyrównana do lewej względem nadrzędnego elementu <ul>, przy czym odległość od lewej krawędzi rośnie o 20% dla każdej kolejnej zakładki.

Na koniec jeszcze raz ustawiamy właściwości wszystkich części zakładek, z tym, że tym razem skupiamy się na stylach zakładki odpowiadającej aktualnie załadowanej stronie oraz na dodaniu efektu podświetlenia po najechaniu na zakładkę kursorem.

```
ul#tabs li:hover,
body#wrox li#tab1,
body#amazon li#tab2,
body#google li#tab3,
body#slashdot li#tab4,
body#twit li#tab5 {
    background: transparent url('images/tab-hover/tab-hover_01.png') no-repeat scroll
left;
}
ul#tabs li:hover > div,
body#wrox li#tab1 > div,
body#amazon li#tab2 > div,
body#google li#tab3 > div,
body#slashdot li#tab4 > div,
body#twit li#tab5 > div {
    background: transparent url('images/tab-hover/tab-hover_03.png') no-repeat scroll
right;
}
ul#tabs li:hover > div > div,
body#wrox li#tab1 > div > div,
body#amazon li#tab2 > div > div,
body#google li#tab3 > div > div,
body#slashdot li#tab4 > div > div,
body#twit li#tab5 > div > div {
    background: transparent url('images/tab-hover/tab-hover_02.png') repeat-x scroll
center;
}
```

Tym razem jedyne, co trzeba zmienić, to obrazek tła. Technika ta, wykorzystując kaskadowy charakter arkuszy, pozwala na zmianę obrazka tła zakładki w przypadku załadowania przypisanej jej strony lub w przypadku najechania na nią kursorem.

Do każdego dokumentu dodaliśmy także wyrażenie warunkowe, widoczne tylko dla Internet Explorera, wprowadzające dodatkowy arkusz stylów przeznaczony wyłącznie dla tej przeglądarki:

```
<!--[if lt IE 7]>
    <link rel='stylesheet' type='text/css' href='tabs-ie.css' />
    <script src='/ie7/ie7-standard-p.js' type='text/javascript'></script>
<![endif]-->
```

Ten arkusz stylów został dołączony za pomocą komentarza warunkowego odczytywanego tylko przez przeglądarki Microsoftu, a który służy do dołączania arkuszy stylów działających poprawnie z tymi przeglądarkami.

Wewnątrz tego arkusza wpisaliśmy tylko jedną regułę:

```
ul#tabs {
    height: 22px;
}
```

Reguła ta likwiduje jednopikselową przestrzeń pod zakładkami.

Po tym, jak nauczyliśmy się implementowania zakładek na naszych stronach na cztery różne sposoby, możemy przejść do następnego rozdziału, w którym pokażę, jak stworzyć wielokolumnowy układ strony przy użyciu XHTML i CSS.