Technologia i rozwiązania

Responsive Web Design
Projektowanie elastycznych witryn w HTML5 i CSS3

Poznaj optymalne przepisy na CakePHP!

Helion

Ben Frain

PACKT PUBLISHING

Tytuł oryginału: Responsive Web Design with HTML5 and CSS3

Tłumaczenie: Maciej Reszotnik

ISBN: 978-83-246-6901-1

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/resweb.zip

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/resweb
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

3

ROZDZIAŁ

Opanowanie układów płynnych

Gdy w latach 90. ubiegłego wieku zacząłem konstruować witryny, ich układ powstawał na bazie tabeli. Najczęściej wymiary pojedynczych sekcji każdej strony były wyrażane w procentach. Przykładowo, lewa kolumna nawigacji mogła wypełniać 20 procent strony, podczas gdy obszar treści 80 procent. W przeszłości nie było tak wielkich różnic między obszarami operacyjnymi przeglądarek jak obecnie, więc te układy były dobrze skalowane w wąskim wyborze ekranów. Nikogo nie obchodziło, że pojedyncze zdania wyglądały odrobinę inaczej na różnych ekranach. Jednakże z biegiem czasu układy CSS stały się normą, bo pozwalały na odzwierciedlenie zasad typograficznych stosowanych w materiałach drukowanych. Zmiana ta oznaczała znaczne zmniejszenie liczby układów konstruowanych na podstawie proporcji — zamiast nich powstawały ich sztywne odpowiedniki, których wymiary definiowano w pikselach.

Podobnie jak inne ważne osiągnięcia, aktualnie ten rodzaj projektowania przeżywa renesans. W przeciągu ostatnich lat miniaturowe samochody, trwała fryzura (chciałoby się…) oraz jeansy dzwony powróciły do łask. Nadszedł czas, by również układy proporcjonalne zostały odkryte na nowo.

W tym rozdziale:

- dowiesz się, czemu układy proporcjonalne są tak ważne w filozofii RWD,
- nauczysz się wyrażać elementy zbudowane w stałej szerokości wartościami procentowymi,
- nauczysz się definiować elementy typografii wyrażone w pikselach w jednostkach `em`,
- zrozumiesz, jak umieścić elementy we właściwych kontekstach,
- nauczysz się płynnie przeskalowywać obrazy,
- dowiesz się, jak dopasować wielkość grafik do wymiarów ekranu,

- zrozumiesz, w jaki sposób uzupełniają się zapytania medialne i płynne układy stron oraz grafik,
- stworzysz układ skalowalny od podstaw w oparciu o system siatek CSS.

Układy stałe nie są przystosowane do nowych wyzwań

Jak już wspomniałem, od czasów układów tabelarycznych nie miałem zbyt wielu okazji, by wykorzystywać projekty proporcjonalne. Zwykle klienci proszą mnie o stworzenie układu najbliżej odzwierciedlającego graficzny model w obrębie 950 – 1000 pikseli. Jeżeli zbudowałbym projekt w oparciu o proporcjonalne wartości (np. 90%), szybko dostałbym skargę: „Ale to wygląda inaczej na moim monitorze!". Witryny skonstruowane na bazie stałych wymiarów wyrażonych w pikselach pozwalały na łatwe dopasowanie układu do prototypu.

Nawet w dzisiejszych czasach, gdy używało się zapytań medialnych celem stworzenia ulepszonej wersji układu, dopasowującej się do szerokości popularnych urządzeń, np. iPada lub iPhone'a (o czym przekonałeś się w rozdziale 2., „Zapytania medialne: obsługa zróżnicowanych obszarów operacyjnych"), wymiary układu w dalszym ciągu mogły być wyrażone w pikselach, ponieważ znaliśmy wielkość obszaru operacyjnego. Choć znajdą się tacy, którzy chętnie wyślą klientowi rachunek za każdym razem, gdy najdzie go ochota, by dodać nowy bajer do strony, nie jest to podejście dobrze przygotujące witrynę na przyszłość. Na rynku pojawiają się coraz to nowsze typy obszarów operacyjnych, więc musimy znaleźć jakiś sposób, by przygotować się na nadejście nieznanego.

Czemu układy proporcjonalne są tak ważne w metodologii RWD?

Mimo że zapytania medialne są bardzo przydatne, mają pewne ograniczenia. Dowolny stały projekt witryny korzystający z zapytań medialnych do dopasowania się do różnych obszarów operacyjnych będzie po prostu przechodził z jednego zestawu reguł do drugiego, bez żadnej animacji pomiędzy stanami. Jak pamiętasz, w przykładzie z rozdziału 2., „Zapytania medialne: obsługa zróżnicowanych obszarów operacyjnych", obszar operacyjny zmieniał się w obrębie stałych zakresów zapytań i wymagał przewijania strony w poziomie dla niestandardowych rozdzielczości (które mogą obowiązywać w nowych, nieznanych dotąd urządzeniach). My jednak chcemy, by nasz układ przewijał się i wyglądał dobrze we wszystkich obszarach operacyjnych, a nie tylko tych zdefiniowanych w zapytaniu medialnym. *Cięcie!* (Rozumiesz? Cięcie! Takie filmowe powiedzenie w sam raz dla naszej filmowej witryny… Prawda? Nie? Już idę po swój płaszcz…). Musimy przemienić nasz stały układ wyrażony w pikselach w układ płynny zbu-

dowany w oparciu o proporcjonalne jednostki. Pozwoli to elementom dynamicznie dopasowywać się względem obszaru operacyjnego do momentu, gdy do akcji wkroczy zapytanie medialne i zmodyfikuje układ.

Symbioza układu proporcjonalnego i zapytań medialnych

Wspomniałem wcześniej artykuł autorstwa Ethana Marcotte'a na temat metodologii RWD, który opublikowano w witrynie A List Apart (*http://www.alistapart.com/articles/responsive-web-design/*). Choć narzędzia, których użył w omawianym przykładzie (płynny układ, obrazy i zapytania medialne), nie były wcale nowe, prawdziwą sensacją było nowatorskie wykorzystanie wszystkich tych technik do stworzenia spójnej całości. Dla wielu projektantów witryn ten artykuł był kluczem do świata nowych możliwości. Przedstawiał sposób, który czerpie to, co najlepsze, z dwóch światów: stworzenie płynnego układu bazującego na projekcie proporcjonalnym przy jednoczesnym ograniczeniu rozstawienia elementów za pomocą zapytania medialnego. Wykorzystanie tych aspektów razem jest fundamentem metodologii RWD, dzięki której powstaje coś znacznie większego niż suma części składowych.

Transformacja stałego układu w projekt proporcjonalny

W najbliższej przyszłości wszystkie gotowe kompozyty, które otrzymasz lub stworzysz, będą miały stałą szerokość. Aktualnie w oprogramowaniu w rodzaju Adobe Photoshop i Fireworks wymiary elementów, wielkość marginesów itp. wyrażamy w pikselach. Dane te wprowadzamy później w arkuszach stylów CSS. To samo dotyczy rozmiarów tekstu. W naszym ulubionym programie graficznym klikamy fragment tekstu, zapisujemy wielkość fontu (też wyrażoną w pikselach) i przenosimy ją do odpowiednich reguł CSS. W jaki więc sposób możemy przemienić nasz układ stały w proporcjonalny?

Ważne równanie

Być może w Twoich oczach jestem zbyt wielkim fanem Ethana Marcotte'a, lecz czuję, że muszę po raz kolejny uchylić mu kapelusza (lub oddać kolejny ukłon w jego stronę albo paść przed nim na kolana). W książce autorstwa Dana Cederholma — *Handcrafted CSS* — Ethan Marcotte współtworzył rozdział poświęcony płynnym siatkom projektowym. W nim opisał prosty i niezawodny wzór służący do transformacji wartości pikselowych w układzie stałym na wartości procentowe w projekcie proporcjonalnym:

```
element docelowy : kontekst = wynik
```

Równanie to jest niepozorne, czyż nie? Szybko jednak okaże się, że jest ono Twoim najlepszym przyjacielem. Zamiast rozprawiać o teoriach, wykorzystajmy ten wzór w praktyce, zamieniając jednostki wymiarów witryny *Zwycięzcą nie jest…* na procenty.

Jak pamiętasz, w rozdziale 2., „Zapytania medialne: obsługa zróżnicowanych obszarów operacyjnych", ustanowiliśmy strukturę kodu naszej strony:

```
<div id="wrapper">
   <!-- nagłówek i nawigacja -->
   <div id="header">
      <div id="navigation">
         <ul>
            <li><a href="#">nawigacja1</a></li>
            <li><a href="#">nawigacja2</a></li>
         </ul>
      </div>
   </div>
   <!-- pasek boczny -->
   <div id="sidebar">
      <p>obszar paska bocznego</p>
   </div>
   <!-- treści  -->
   <div id="content">
      <p>obszar głównej zawartości strony</p>
   </div>
   <!-- stopka -->
   <div id="footer">
      <p>stopka</p>
   </div>
</div>
```

W dalszej części dodaliśmy treści, ale tym, co nas interesuje w tej chwili, są wymiary elementów strukturalnych (nagłówka, nawigacji, paska bocznego, treści i stopki) w arkuszach CSS. Niżej pominąłem wiele reguł formatowania, byśmy mogli skupić naszą uwagę na strukturze:

```
#wrapper {
   margin-right: auto;
   margin-left: auto;
   width: 960px;
}
#header {
   margin-right: 10px;
   margin-left: 10px;
   width: 940px;
}
#navigation {
   padding-bottom: 25px;
   margin-top: 26px;
   margin-left: -10px;
   padding-right: 10px;
   padding-left: 10px;
   width: 940px;
}
#navigation ul li {
```

```
display: inline-block;
}
   #content {
   margin-top: 58px;
   margin-right: 10px;
   float: right;
   width: 698px;
}
#sidebar {
   border-right-color: #e8e8e8;
   border-right-style: solid;
   border-right-width: 2px;
   margin-top: 58px;
   padding-right: 10px;
   margin-right: 10px;
   margin-left: 10px;
   float: left;
   width: 220px;
}
#footer {
   float: left;
   margin-top: 20px;
   margin-right: 10px;
   margin-left: 10px;
   clear: both;
   width: 940px;
}
```

Wszystkie wymiary są obecnie wyrażone w pikselach. Zacznijmy od elementu położonego najbardziej na zewnątrz struktury układu i zmieńmy jednostki jego wymiarów na procenty, korzystając ze wzoru *element docelowy : kontekst = wynik*.

Wszystkie nasze treści są zawarte w elemencie `div`, oznaczonym identyfikatorem `#wrapper`. Z kodu CSS łatwo odczytać, że jego marginesy są ustawione automatycznie i sam element ma 960 pikseli. Jak zdefiniować, jak wielką część obszaru operacyjnego powinien zajmować taki zewnętrzny komponent `div`?

Definiowanie kontekstu w elementach proporcjonalnych

Potrzebujemy czegoś, co pomieści wszystkie proporcjonalne elementy (treści, pasek boczny, stopkę itd.), stając się dla nich kontekstem, który zawrzemy w naszym projekcie. Z tego powodu musimy wyznaczyć proporcjonalną wartość szerokości komponentu `#wrapper` w stosunku do wymiarów obszaru operacyjnego. Na razie przeprowadźmy eksperyment, ustawiając szerokość na 96 procent, i zobaczmy, co się stanie. Oto poprawiona reguła komponentu `#wrapper`:

```
#wrapper {
   margin-right: auto;
   margin-left: auto;
   width:  96%;  /* Dotyczy położonego najbardziej na zewnątrz elementu DIV */
}
```

A oto jak nasza strona prezentuje się w oknie przeglądarki:

Na razie nie jest źle! Wartość 96 procent sprawdza się całkiem dobrze, choć moglibyśmy wybrać 100 albo 90 procent — dowolną wielkość, która wyznaczyłaby wygodne granice w obszarze operacyjnym.

Schodząc w dół struktury witryny, szybko zauważymy, że zmiana układu na proporcjonalny staje się coraz bardziej skomplikowana. Najpierw przyjrzyjmy się strukturze nagłówka. Przypomnij sobie wzór: *element docelowy : kontekst = wynik*. Nasz element docelowy (`#header div`) znajduje się wewnątrz elementu `div #wrapper` (kontekstu). Dlatego ustalamy, że `#header` (element docelowy) ma szerokość 940 pikseli, po czym dzielimy tę wartość przez szerokość kontekstu (komponentu `#wrapper`), która miała wartość 960 pikseli, i otrzymujemy wynik 0,979166667. Wartość tę wyrażamy w procentach, przesuwając dwie pierwsze cyfry ułamka przed przecinek dziesiętny, i otrzymujemy wartość procentową 97,9166667. Zapiszmy ją w regule CSS:

```
#header {
   margin-right: 10px;
   margin-left: 10px;
   width: 97,9166667%; /*  940 : 960 */
}
```

Zauważmy, że elementy `#navigation` i `#footer` mają tę samą szerokość, więc bez problemu możemy zamienić wartości wyrażone w pikselach na procenty.

Na koniec, zanim otworzymy stronę w przeglądarce, zajmijmy się elementami `#content` i `#wrapper`. Z uwagi na fakt, iż kontekst pozostał ten sam (960 pikseli), wystarczy, że podzielimy szerokość elementu docelowego przez jego rozpiętość. Nasz element `#content` ma aktualnie 698 pikseli, więc podzielmy to przez 960, a otrzymamy wynik równy 0,727083333. Przesuńmy przecinek dziesiętny o dwa miejsca, a otrzymamy 72,7083333 procenta — jest to nowa, procentowa szerokość komponentu `div` z identyfikatorem `#content`. Z kolei nasz panel boczny ma szerokość 220 pikseli, lecz musimy też pamiętać o szerokiej na 2 piksele krawędzi. Nie chcemy, by grubość prawej krawędzi rozszerzała się lub zwężała, więc w dalszym ciągu będzie ona miała 2 piksele. Z tego względu należy ująć odrobinę z szerokości paska bocznego. Odjąłem więc 2 piksele i przeprowadziłem obliczenia według naszego równania. Podzieliłem szerokość elementu docelowego (czyli po poprawce 218 pikseli) przez rozpiętość kontekstu (960 pikseli) — otrzymałem wynik 0,227083333. Przenoszę przecinek dziesiętny o dwa miejsca i dostaję 22,7083333 procenta szerokości paska bocznego. Po zmodyfikowaniu wszystkich wartości nasz arkusz CSS prezentuje się w następujący sposób:

```
#wrapper {
   margin-right: auto;
   margin-left: auto;
   width: 96%; /* Dotyczy położonego najbardziej na zewnątrz elementu DIV */
}
#header {
   margin-right: 10px;
   margin-left: 10px;
   width: 97.9166667%; /* 940 : 960 */
}
#navigation {
   padding-bottom: 25px;
```

```
    margin-top: 26px;
    margin-left: -10px;
    padding-right: 10px;
    padding-left: 10px;
    width: 72.7083333%; /* 698 : 960 */
}
#navigation ul li {
    display: inline-block;
}
#content {
    margin-top: 58px;
    margin-right: 10px;
    float: right;
    width: 72.7083333%; /* 698 : 960 */
}
#sidebar {
    border-right-color: #e8e8e8;
    border-right-style: solid;
    border-right-width: 2px;
    margin-top: 58px;
    margin-right: 10px;
    margin-left: 10px;
    float: left;
    width: 22.7083333%; /* 218 : 960 */
}
#footer {
    float: left;
    margin-top: 20px;
    margin-right: 10px;
    margin-left: 10px;
    clear: both;
    width: 97.9166667%; /* 940 : 960 */
}
```

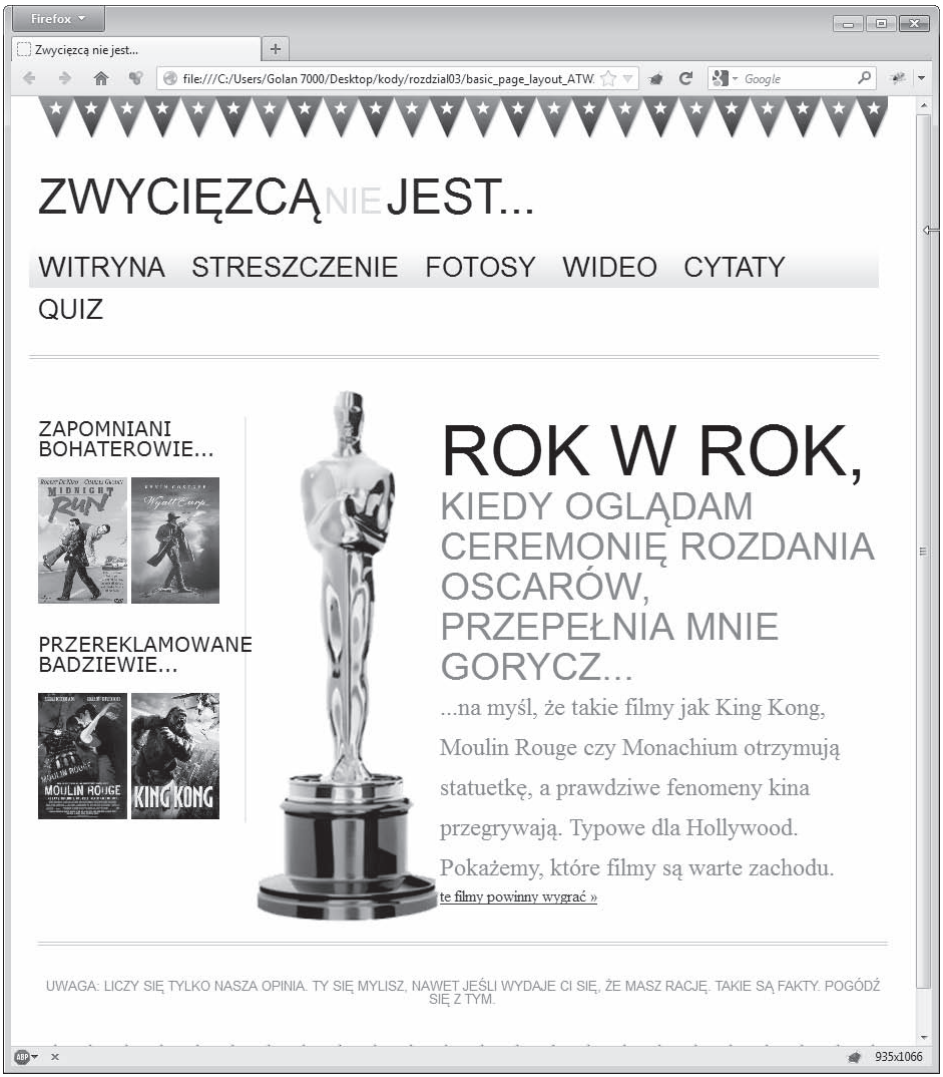
Poniższy zrzut ekranu przedstawia wygląd strony w oknie przeglądarki Firefox, którego obszar operacyjny ma szerokość powyżej 1000 pikseli:

Dobrze nam idzie. Zastąpmy teraz wszystkie deklaracje `10 px` właściwości `padding` i `margin` w arkuszu wartościami wziętymi ze wzoru *element docelowy : kontekst = wynik*. Wszystkie komponenty o szerokości 10 pikseli mają ten sam kontekst, a ich rozpiętość wyrażona w procentach wynosi 1,0416667.

A tych liczb to nie można zaokrąglić?

Niektórzy krytycy metodologii RWD (np. autor artykułu *http://tripleodeon.com/2010/10/not-a-mobile-web-merely-a-320px-wide-one/*) twierdzą, że wprowadzanie liczb w rodzaju `0.550724638 em` wewnątrz arkusza stylów jest śmieszne. Możesz sam się zastanawiać, czy nie warto byłoby ich zaokrąglić? Odpowiedź brzmi: przeglądarkę należy informować z jak największą dokładnością. Dzięki temu będzie ona w stanie precyzyjniej wyświetlać wszystkie elementy. Tak na marginesie, jeśli w szkole nie zasypiałeś zbyt często na lekcjach matematyki, z pewnością słyszałeś o złotej proporcji (*http://pl.wikipedia.org/wiki/Złoty_podział*). Ta matematyczna proporcja charakteryzuje każdą dyscyplinę naukową i wynosi 1:1,61803398874989 (jeśli chcesz poznać jej 10 000 miejsce po przecinku, zajrzyj na stronę *http://www.maths.surrey.ac.uk/hosted-sites/R.Knott/Fibonacci/phi10000dps.txt*). Nie jest to równa liczba, lecz ma ona wielkie znaczenie. Jeśli złota proporcja może być wyrażona w taki sposób, czemu nie nasze projekty witryn?

Wszystkie elementy w dalszym ciągu prezentują się bardzo dobrze w tym samym rozmiarze obszaru operacyjnego. Jedyny problem stanowi nawigacja. Jeśli odrobinę zmniejszę szerokość obszaru operacyjnego, linki rozwiną się w dwóch wierszach:



Co więcej, jeśli rozszerzę okno, margines pomiędzy linkami nie będzie zwiększał się proporcjonalnie. Rzućmy okiem na reguły CSS definiujące nawigację i spróbujmy ustalić, czemu tak się dzieje:

```
#navigation {
   padding-bottom: 25px;
   margin-top: 26px;
```

```
    margin-left: -1.0416667%; /* 10 : 960 */
    padding-right: 1.0416667%; /* 10 : 960 */
    padding-left: 1.0416667%; /* 10 : 960 */
    width: 97.9166667%; /* 940 : 960 */
    background-repeat: repeat-x;
    background-image: url(../img/atwiNavBg.png);
    border-bottom-color: #bfbfbf;
    border-bottom-style: double; border-bottom-width: 4px;
}
#navigation ul li {
    display: inline-block;
}
#navigation ul li a {
    height: 42px;
    line-height: 42px;
    margin-right: 25px;
    text-decoration: none;
    text-transform: uppercase;
    font-family: Arial, "Lucida Grande", Verdana, sans-serif;
    font-size: 27px;
    color: black;
}
```

Na pierwszy rzut oka wydaje się, że za cały chaos jest odpowiedzialna trzecia reguła dla selektora `#navigation ul li a`, który ma margines równy 25 pikseli. Naprawmy to, stosując nasze równanie. Element `#navigation` miał szerokość 940 pikseli, więc ostateczny wynik procentowy jest równy 2,6595745. Zmieńmy więc naszą regułę w następujący sposób:

```
#navigation ul li a {
    height: 42px;
    line-height: 42px;
    margin-right: 2.6595745%; /* 25 : 940 */
    text-decoration: none;
    text-transform: uppercase;
    font-family: Arial, "Lucida Grande", Verdana, sans-serif;
    font-size: 27px;
    color: black;
}
```

To nie było takie trudne! Sprawdźmy, co pokazuje nam nasza przeglądarka…

A niech to! Nie o to nam chodziło. Linki nie rozchodzą się już w dwóch wierszach, lecz nie znamy właściwej proporcji marginesów. Linki nawigacyjne zlewają się w jedno długie słowo, którego próżno szukać w słowniku.

Zawsze należy pamiętać o kontekście

Analizując uważnie nasz wzór (*element docelowy : kontekst = wynik*), łatwo zrozumieć, czemu tak się dzieje. Naszym problemem w tym miejscu jest kontekst. Rzuć okiem na kod:

```
<div id="navigation">
   <ul>
      <li><a href="#">O witrynie</a></li>
      <li><a href="#">Streszczenie</a></li>
      <li><a href="#">Fotosy</a></li>
      <li><a href="#">Wideo</a></li>
```

```
        <li><a href="#">Cytaty</a></li>
        <li><a href="#">Quiz</a></li>
    </ul>
</div>
```

Jak zauważysz, elementy `<a href="#">` znajdują się wewnątrz znacznika `<li>`. To one są kontekstem dla naszego proporcjonalnego marginesu. Gdy przyjrzymy się regule CSS tych znaczników, okaże się, że nie zdefiniowaliśmy dla nich szerokości:

```
#navigation ul li { display: inline-block; }
```

Jak większość problemów, tak i ten można rozwiązać na wiele sposobów. Moglibyśmy zadeklarować pewną szerokość dla elementu `<li>`, ale musielibyśmy wyrazić ją w pikselach lub w procentach nadrzędnego komponentu (`#navigation`) — żadne z tych rozwiązań nie jest dość elastyczne, by odpowiednio sformatować tekst, który pojawia się w nawigacji.

Zamiast tego moglibyśmy poprawić znacznik `<li>`, zmieniając właściwość `display` na `inline`.

```
#navigation ul li {
    display: inline;
}
```

Wybór opcji `display: inline;` (która sprawia, że elementy `<li>` tracą swój blokowy status) powoduje, że komponenty nawigacji są wyświetlane w poziomie nawet we wcześniejszych wersjach Internet Explorera (6 i 7), które miały problemy ze zinterpretowaniem właściwości `inline-block`. Muszę jednak przyznać, że jestem wielkim zwolennikiem własności `inline-block`, ponieważ daje ona większą kontrolę nad marginesami i wypełnieniem w nowych przeglądarkach, więc na razie zostawię ją bez zmian (być może później dodam style dla IE 6 i 7), a zamiast tego przeniosę regułę marginesu ze znacznika `<a>` (który nie ma jasnego kontekstu) do bloku `<li>`. Oto jak prezentują się reguły po poprawkach:

```
#navigation ul li {
    display: inline-block;
    margin-right: 2.6595745%;  /* 25  :  940 */
}
#navigation ul li a {
    height: 42px;
    line-height: 42px;
    text-decoration: none;
    text-transform: uppercase;
    font-family: Arial, "Lucida Grande",
    Verdana, sans-serif;
    font-size: 27px;
    color: black;
}
```

Poniższy zrzut ekranu pokazuje, jak strona wygląda w przeglądarce o obszarze operacyjnym szerokim na 1200 pikseli:

Nawigacja jest już prawie gotowa, lecz nadal mamy problem z jej zawijaniem się do dwóch linijek, w momencie zmniejszenia się obszaru operacyjnego do wielkości 768 pikseli, gdy zapytanie medialne, które przygotowaliśmy w rozdziale 2., „Zapytania medialne: obsługa zróżnicowanych obszarów operacyjnych", nadpisze aktualne style nawigacji. Nim przejdziemy do jej naprawienia, zmienię wszystkie jednostki wielkości elementów typografii z pikseli na em. Gdy to się nam powiedzie, przyjrzymy się kolejnemu słoniowi w składzie porcelany, czyli kwestii skalowania obrazów wraz z projektem.

Wykorzystywanie jednostek em zamiast pikseli w kontekście typografii

Dawno, dawno temu projektanci witryn wykorzystywali głównie jednostki `em` zamiast pikseli do definiowania elementów typograficznych, ponieważ wczesne wersje Internet Explorera nie były w stanie przybliżać tekstu zdefiniowanego w pikselach. Od jakiegoś czasu nowoczesne przeglądarki potrafią ominąć tę niedogodność. Czemu więc używanie jednostek `em` jest lepszym, jeśli nie wymaganym rozwiązaniem? Są ku temu dwa oczywiste powody: po pierwsze, wszystkie osoby używające Internet Explorera 6 (tak, te dwa indywidua) automatycznie otrzymują opcję przybliżania tekstu; po drugie, znacznie ułatwia to Twoją pracę. Wartość jednostki `em` zależy od rozmiarów jej kontekstu. Jeśli zdefiniujemy wielkość fontu w całym elemencie body i ustalimy reguły typograficzne, używając jednostek `em`, nasza pierwsza deklaracja będzie miała wpływ na wszystkie style. Zaletą jest fakt, że przy prawidłowych ustawieniach, jeśli klient poprosi Cię o powiększenie fontów, wystarczy zmienić parametry deklaracji dla elementu body, by typografia w całej witrynie uległa modyfikacji.

Używając znanego nam równania *element docelowy : kontekst = wynik*, zamierzam zamienić każdą wartość wyrażoną w pikselach na wartość `em`. Warto zauważyć, że we wszystkich nowoczesnych przeglądarkach wartość `font-size` domyślnie jest równa 16 pikseli (chyba że ją sam nadpisałeś). Z tego powodu wykorzystanie dowolnej z poniższych reguł do znacznika `body` zaowocuje takim samym wynikiem:

```
font-size: 100%;
font-size: 16px;
font-size: 1em;
```

Aby lepiej zrozumieć ten mechanizm, zwróć uwagę na definicję wielkości fontu w tytule witryny *ZWYCIĘZCĄ NIE JEST…* u góry po lewej stronie:

```
#logo {
   display: block;
   padding-top: 75px;
   color: #0d0c0c;
   text-transform: uppercase;
   font-family: Arial, "Lucida Grande", Verdana, sans-serif;
   font-size: 48px;
}
#logo span { color: #dfdada; }
```

Podzielmy więc: *48 : 16 = 3*. Nasze style mają więc następującą postać:

```
#logo {
   display: block;
   padding-top: 75px;
   color: #0d0c0c;
```

```
    text-transform: uppercase;
    font-family: Arial, "Lucida Grande", Verdana, sans-serif;
    font-size: 3em; /* 48 : 16 = 3 */
}
```

Ten sam mechanizm sprawdzi się w każdym przypadku. Jeśli w którymś momencie na stronie zapanuje chaos, może to wynikać ze zmiany w kontekście. Dla przykładu spójrz na ten oto fragment strony:

```
<h1>Rok w rok, <span>kiedy oglądam ceremonię rozdania Oscarów,
↪przepełnia mnie gorycz...</span></h1>
```

Nasz nowy układ bazujący na jednostkach `em` został zdefiniowany w CSS następująco:

```
#content h1 {
    font-family: Arial, Helvetica, Verdana, sans-serif;
    text-transform: uppercase;
    font-size: 4.3125em; } /* 69 : 16 */
#content h1 span {
    display: block;
    line-height: 1.052631579em; /* 40 : 38 */
    color: #757474;
    font-size: .550724638em; /* 38 : 69 */
}
```

Zwróć uwagę na zadeklarowaną wielkość fontu elementu `<span>` (`38 px`) w stosunku do deklaracji w elemencie rodzica (`69 px`). Zauważ też, że właściwość `line-height` (równa tu `40 px`) zależy od wielkości samego fontu (czyli `38 px`).

em — co to takiego?

Jednostka *em* wzięła się ze sposobu wymawiania litery „M", który został przeniesiony na formę pisemną. Dawniej litera „M" była wykorzystywana do ustalenia wielkości danego fontu, ponieważ jest największą (i najszerszą) z liter. Obecnie jednostka em oznacza proporcję szerokości i wysokości danej litery w odniesieniu do wielkości danego fontu.

Struktura naszej strony jest już elastyczna i udało nam się zmienić wszystkie jednostki wielkości kroju pisma na `em`. W dalszym ciągu jednak nie wiemy, jak przeskalować obrazy wraz ze zmianą wielkości obszaru operacyjnego, więc warto się tym teraz zająć.

Płynne obrazy

Łatwo dokonać przeskalowania obrazów w płynnym układzie strony w nowoczesnych przeglądarkach (w tym w IE 7+). Wystarczy tylko zadeklarować następującą regułę CSS:

```
img {
   max-width: 100%;
}
```

W rezultacie wszystkie obrazy wypełnią 100 procent zawierającego je elementu. Co więcej, ten sam atrybut i właściwość sprawdzą się w skalowaniu innych typów medialnych:

```
img,object,video,embed {
   max-width: 100%;
}
```

Większość z nich stanie się skalowalna, z pewnymi istotnymi wyjątkami, jak filmy zagnieżdżone w elemencie `<iframe>`, lecz zajmiemy się nimi później w rozdziale 4., „HTML5 i projekty elastyczne". Na razie skupimy się na obrazach — mechanizm pozostaje ten sam niezależnie od typu multimediów.

Nim przystąpimy do działania, zastanówmy się nad kolejnymi krokami. Musimy zaplanować działania z wyprzedzeniem — wstawiane obrazy powinny być dość duże, by zmieścić się w obrębie szerokiego obszaru operacyjnego. To zmusza nas do rozważenia jeszcze jednej kwestii. Niezależnie od szerokości obszaru operacyjnego użytkownicy będą musieli ściągnąć duży obraz, nawet jeśli ekran będzie w stanie pomieścić jedynie 25 procent grafiki. Jest to ważne zagadnienie z punktu widzenia przepustowości łącza, więc wkrótce poświęcimy mu więcej uwagi. Na razie skoncentrujmy się na skalowaniu obrazów.

Skalowanie obrazów w obrębie obszaru operacyjnego

Spójrz na nasz pasek boczny zawierający plakaty dwóch fantastycznych filmów i dwóch kiczowatych szmir (nawet nie próbuj o tym dyskutować). Kod struktury paska aktualnie prezentuje się następująco:

```
<!-- pasek boczny -->
<div id="sidebar">
   <div class="sideBlock unSung">
      <h4>Zapomniani bohaterowie...</h4>
      <a href="#"><img src="img/midnightRun.jpg" alt="midnight_run"
      ↪width="99" height="135" /></a>
      <a href="#"><img src="img/wyattEarp.jpg" alt="wyatt-earp" width="99"
      ↪height="135" /></a>
   </div>
   <div class="sideBlock overHyped">
      <h4>Przereklamowane badziewie...</h4>
```

```
            <a href="#"><img src="img/moulinRouge.jpg" alt="MoulinRouge"
            ↪width="99" height="135" /></a>
            <a href="#"><img src="img/kingKong.jpg" alt="kingkong" width="99"
            ↪height="135" /></a>
        </div>
</div>
```

Choć dodałem deklarację `max-width: 100%` dla elementu `img` w obszarze operacyjnym, nic się nie zmieniło — obrazy nie są przeskalowywane, gdy powiększam obszar operacyjny.

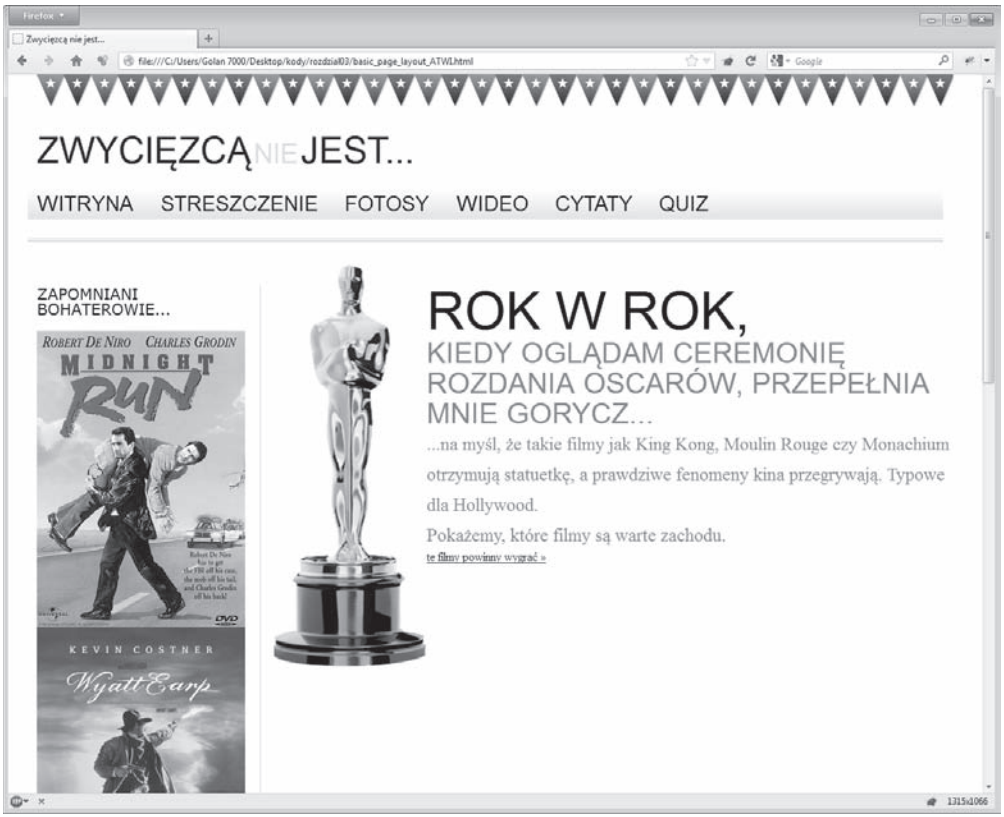
Jest tak dlatego, że w kodzie HTML zadeklarowałem szerokość i wysokość danego obrazu:

```
<img src="img/wyattEarp.jpg" alt="Wyatt Earp" width="99" height="135" />
```

Kolejny szkolny błąd! Poprawiam więc kod, usuwając ze znacznika obrazów atrybuty `height` i `width`.

```
<img src="img/wyattEarp.jpg" alt="Wyatt Earp" />.
```

Zobaczmy, co się stanie, gdy odświeżymy okno:



To działa! Pojawia się jednak kolejny problem. Obrazy są przeskalowywane tak, aby wypełnić 100% szerokości elementu rodzica, więc wypełniają sobą całą szerokość paska bocznego. Jak zawsze, jest wiele metod, żeby to naprawić.

Właściwe reguły dla właściwych obrazów

Mógłbym dodać do każdego elementu obrazu dodatkową klasę, co ma miejsce w poniższym fragmencie:

```
<img class="side!mage" src="img/wyattEarp.jpg" alt="Wyatt Earp" />
```

Teraz wystarczyłoby ustalić w tej klasie określoną szerokość. Zamiast tego wolę pozostawić kod witryny w spokoju i wykorzystać CSS, aby nadpisać już zdefiniowaną własność `max-width` regułą dotyczącą obrazów w pasku bocznym:

```
img {
   max-width: 100%;
}
.sideBlock img {
   max-width: 45%;
}
```

Poniższy rysunek prezentuje wygląd naszej strony po zmianach:

Wykorzystanie tego mechanizmu CSS pozwala uzyskać dodatkową kontrolę nad szerokością obrazów i innych typów multimediów. W rozdziale 5., „CSS3: selektory, typografia i tryby barw" dowiesz się, w jaki sposób nowe selektory CSS3 pozwalają nam odwołać się do dowolnego elementu bez potrzeby stosowania dodatkowego kodu czy korzystania z biblioteki jQuery do wykonania niewdzięcznej roboty.

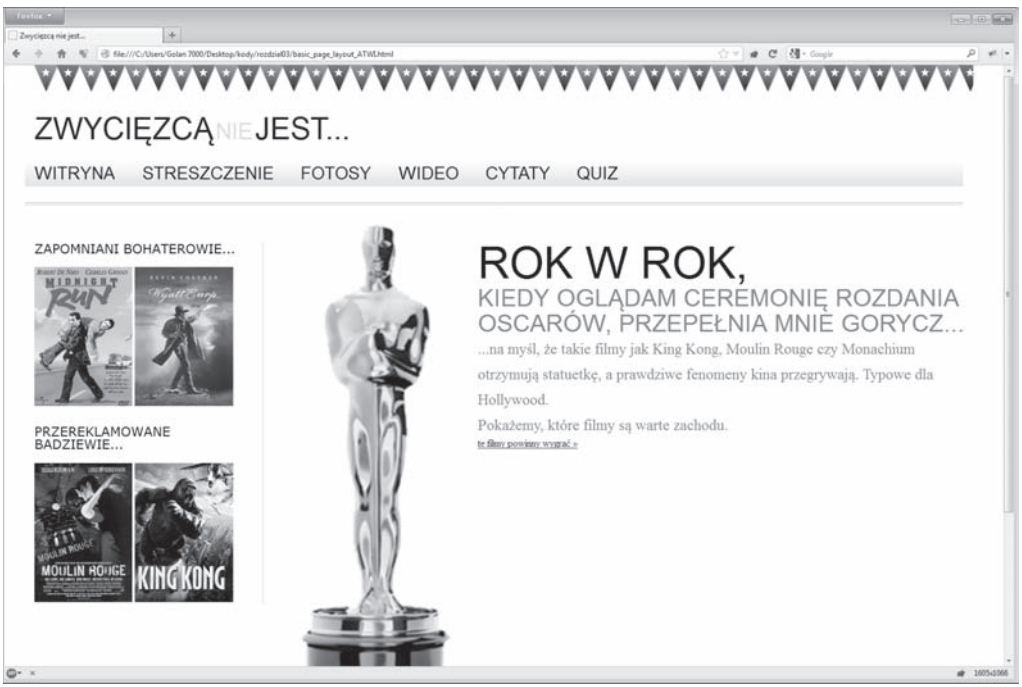
Postanowiłem, że obrazy w panelu bocznym będą miały 45 procent szerokości elementu rodzica, ponieważ wiem, że pomiędzy nimi powinienem wstawić drobny margines — jeśli oba obrazy zajmą 90 procent całkowitej rozpiętości, będę miał wystarczająco dużo miejsca (10 procent), by wykonać zadanie.

Obrazy w pasku bocznym prezentują się znakomicie, więc usuńmy atrybuty `width` i `height` ze statuetki Oscara. Jednakże jeśli nie zdefiniujemy proporcjonalnej wartości parametru `width`, obraz nie będzie się skalować, więc należy ustawić odpowiednie właściwości w arkuszu CSS, według dobrze nam znanej formuły: *element docelowy : kontekst = wynik*.

```
.oscarMain {
   float: left;
   margin-top: -28px;
   width:  28,9398281%;  /* 698 : 202 */
}
```

Nakładanie hamulców na układ płynny

Nasze obrazy skalują się dobrze w odpowiedzi na rozszerzanie i zwężanie obszaru operacyjnego. Jeśli jednak rozszerzymy okno zbyt mocno, naszym oczom ukaże się niezbyt przyjemny widok. Rzuć okiem na statuetkę Oscara w rozdzielczości 1600 px:



Obraz w pliku *oscar.png* ma szerokość 202 pikseli. Jeśli jednak rozszerzymy okno do rozpiętości 1600 pikseli, wyświetlana grafika będzie miała szerokość ponad 270 pikseli. Możemy jednak łatwo „nałożyć hamulce” na obraz, ustanawiając kolejną regułę:

```
.oscarMain {
   float: left;
   margin-top: -28px;
   width: 28,9398281%;   /* 698 : 202 */
   max-width: 202px;
}
```

Reguła ta pozwoli grafice rozszerzać się, lecz nigdy nie przekroczy ona szerokości zdefiniowanej w znaczniku `max-width`. Gdy zdefiniujemy tę wartość, strona będzie prezentować się podobnie, jak na poniższym zrzucie ekranu:

Wszechstronna własność max-width

Kolejnym hamulcem dla nieograniczonego rozszerzania się strony jest nałożenie własności `max-width` na element `#wrapper`:

```
#wrapper {
   margin-right: auto;
   margin-left: auto;
   width: 96%; /* Dotyczy położonego najbardziej na zewnątrz elementu DIV */
   max-width: 1414px;
}
```

W rezultacie układ witryny będzie zajmował 96 procent obszaru operacyjnego, lecz nigdy nie przekroczy granicy 1414 pikseli (zdecydowałem się ustawić ją w tym miejscu, ponieważ w większości przeglądarek kończy to szereg chorągiewek we właściwym momencie, tak że żadna z nich nie jest przecięta w środku). Następujący zrzut ekranu pokazuje, jak nasza strona prezentuje się w obszarze operacyjnym szerokim na 1600pikseli:

Naturalnie to zaledwie kilka opcji. Dowodzą one jednak wszechstronności płynnego układu i pokazują, jak łatwo można kontrolować przepływ strony przy użyciu kilku deklaracji.

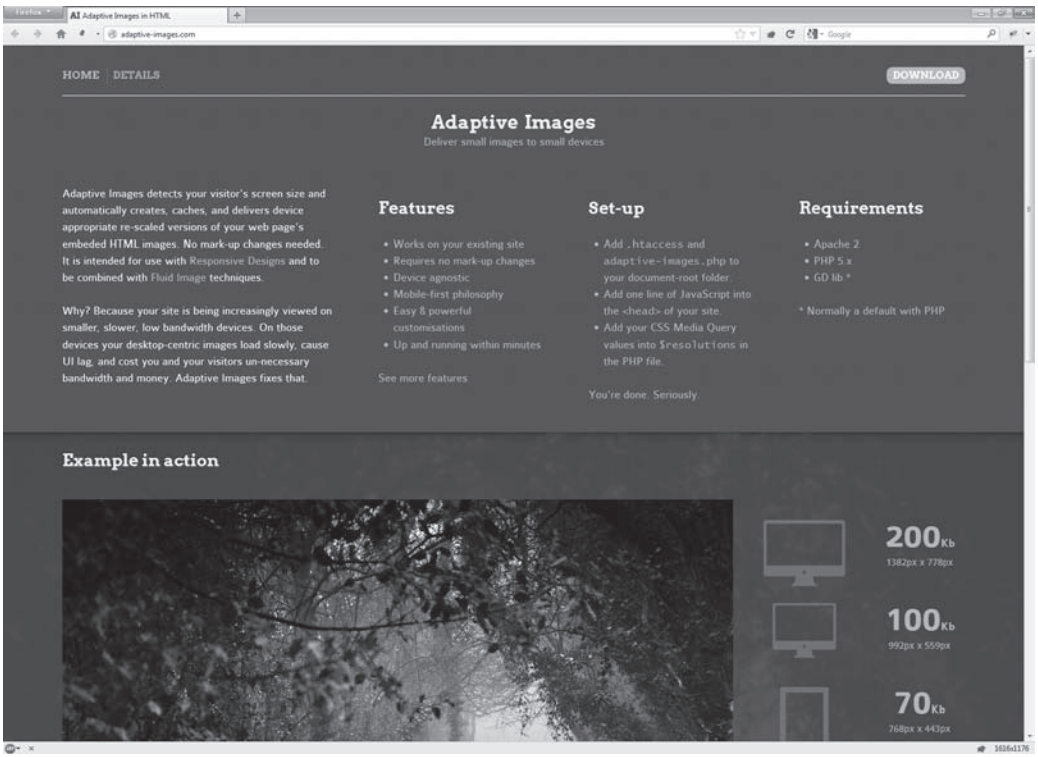
Wczytywanie różnych obrazów dla różnych ekranów

Nasze obrazy są skalowane w oknie przeglądarki i kontrolujemy dokładnie ich wielkość dla rozpiętości okna. Wcześniej w tym rozdziale zauważyliśmy podstawowy problem związany ze skalowaniem grafik. Wielkość ich plików musi być duża, aby były dobrze wyświetlane na stronach. Jeśli nie są, będą wyglądać szkaradnie. Z tego względu pliki obrazów są zwykle większe, niż jest to konieczne, by zostały wczytane w oknie przeglądarki.

Wiele osób próbowało rozwiązać ten problem poprzez wczytanie mniejszych obrazów pasujących do mniejszych wyświetlaczy. Pierwszym wartym uwagi rozwiązaniem jest usługa oferowana przez zespół Filament Group zwana „Responsive Images" (*http://filamentgroup.com/lab/responsive_images_experimenting_with_context_aware_image_sizing/*). Ostatnio jednak zacząłem używać udostępnianego przez Matta Wilcoxa narzędzia „Adaptive Images"(*http://adaptive-images.com*). Rozwiązanie zespołu Filament Group wymagało dokonania zmian w kodzie HTML strony. Rozwiązanie Matta jest bardziej wszechstronne i dodatkowo automatycznie tworzy mniejsze wersje obrazów na podstawie wymiarów zdefiniowanych w kodzie HTML. Pozwala ono na zmniejszenie wielkości obrazów i przesłanie właściwej grafiki w zależności od liczby punktów kontrolnych wymuszających zmianę układu strony. Sprawdźmy więc, ile jest warta usługa Adaptive Images.

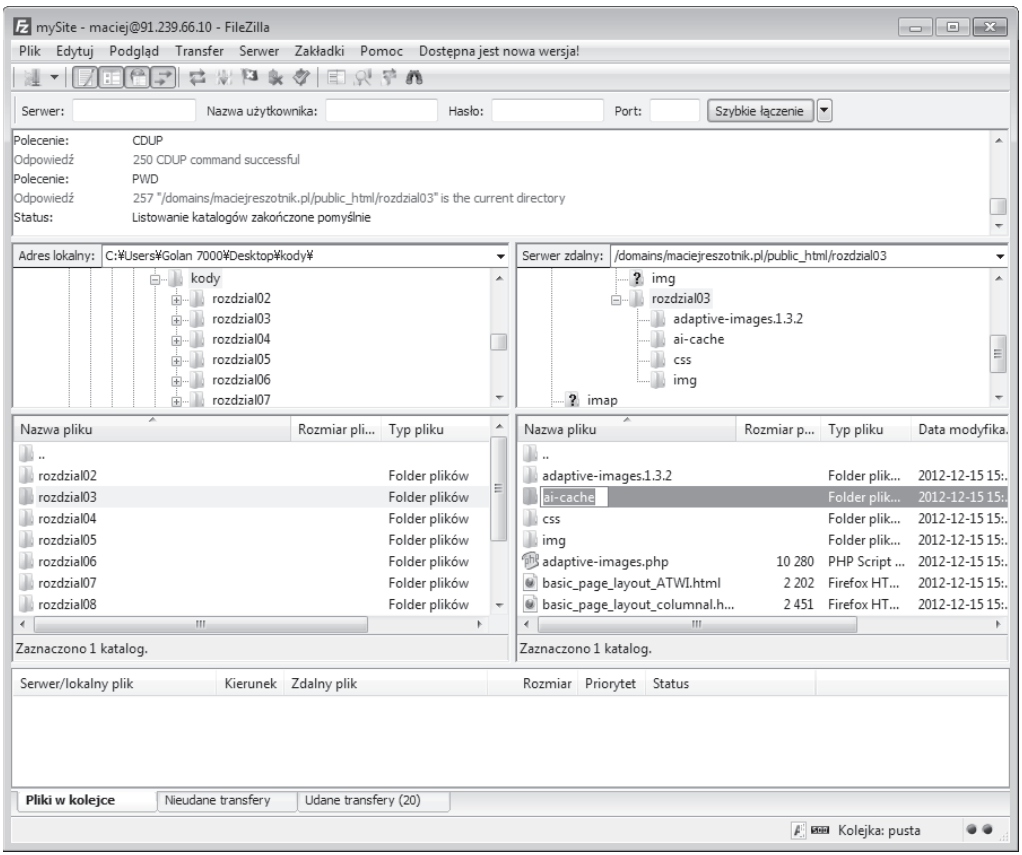
Konfigurowanie usługi Adaptive Images

Rozwiązanie Adaptive Images wymaga serwera Apache 2 z zainstalowanym środowiskiem PHP 5.x oraz biblioteką GD Lib. Musisz więc dysponować odpowiednim serwerem, by skorzystać z oferowanych przez nie funkcji. Pobierz plik *.zip* i przystąpmy do działania.



Rozpakuj archiwum i umieść skrypt *adaptive-images.php* wraz z plikiem *.htaccess* w katalogu głównym witryny. Jeśli już masz plik *.htaccess* w katalogu głównym, nie nadpisuj go. Zamiast tego postępuj zgodnie z instrukcjami z pliku *instructions.htm*, który dodano do archiwum.

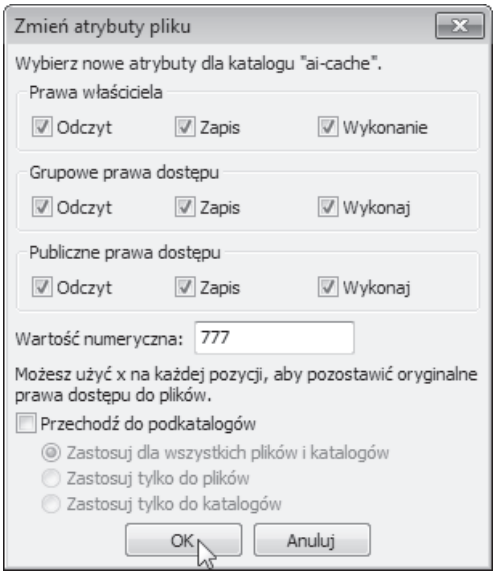
Teraz w katalogu głównym stwórz folder i nadaj mu nazwę *ai-cache*.



Użyj funkcji swojego ulubionego klienta FTP i ustaw prawa dostępu do pliku na 777.

Teraz umieść poniższy fragment kodu JavaScript w komponencie `<head>` wszystkich stron, na których skorzystasz z funkcji adaptacyjnych obrazów.

```
<script> document.cookie='resolution='+Math.max(screen.width,screen.height)+';
➥path=/';</script>
```



Zwróć uwagę, że jeżeli nie używasz HTML5 (standard ten wykorzystamy w następnym rozdziale) i jeżeli chcesz, by strona była automatycznie poddawana walidacji, musisz dodać atrybut `type`. Dlatego też nasz kod powinien prezentować się następująco:

```
<script type="text/javascript">document.cookie='resolution='+Math.max
↪(screen.width,screen.height)+'; path=/';</script>
```

Kod JavaScript musi znajdować się w elemencie `head` (najlepiej jeśli jest to pierwszy element `<script>` w dokumencie), ponieważ powinien zadziałać, zanim strona zostanie w pełni wczytana, przed wysłaniem żądań o przesłanie obrazów. Poniższy przykład pokazuje jego pozycję w komponencie `<head>` strony:

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="viewport"  content="width=device-width,initial-scale=1.0" />
<title>Zwycięzcą nie jest…</title>
<script type="text/javascript">document.cookie='resolution='+Math.
↪max(screen.width,screen.height)+'; path=/';</script>
<link href="css/main.css" rel="stylesheet" type="text/css" />

</head>
```

Przenoszenie obrazów tła do innej lokalizacji

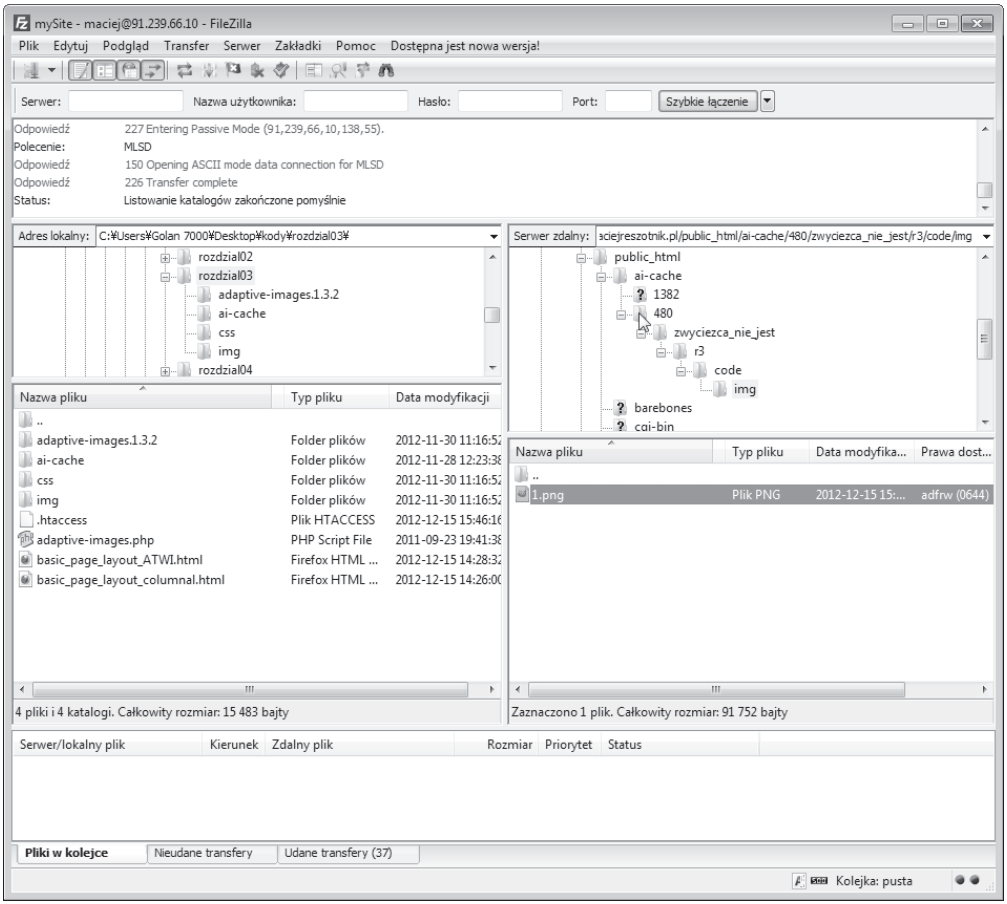
W przeszłości zwykle umieszczałem wszystkie swoje obrazy (zarówno te używane do definiowania tła w CSS jak i grafiki zagnieżdżane w kodzie HTML) w tym samym folderze, który nazywałem *images* lub *img*. Jednak w przypadku narzędzia Adaptive Images lepiej jest, jeśli obrazy tła, na które powołujesz się w CSS (lub inne grafiki, których wielkości nie chcesz modyfikować), zostaną przeniesione do innego katalogu. Narzędzie Adaptive Images domyślnie definiuje odrębny folder *assets*, przeznaczony do przechowywania obrazów, które nie powinny zostać zmniejszone. Jeżeli nie chcesz, by Twoje grafiki zostały zmienione, przenieś je do tego folderu. Aby zdefiniować inny folder docelowy w pliku *.htaccess*, wprowadź następującą modyfikację:

```
<IfModule mod_rewrite.c>
  Options +FollowSymlinks
  RewriteEngine On
  # Adaptive-Images ---------------------------------------------------
  RewriteCond %{REQUEST_URI} !assets
  RewriteCond %{REQUEST_URI} !bkg
  # Wyślij żądanie o przesłanie dowolnego pliku GIF, JPG lub PNG, który
  ↪NIE ZOSTAŁ zachowany w wymienionych wyżej katalogach, do skryptu
  ↪adaptive-images.php, aby wybrał jego wersję mającą właściwy rozmiar.
  RewriteRule \.(?:jpe?g|gif|png)$ adaptive-images.php
  # END Adaptive-Images -----------------------------------------------
</IfModule>
```

W tym przykładzie zadeklarowaliśmy, że nie chcemy, by nasze skalowalne obrazy były odczytywane z katalogów *assets* lub *bkg*. W odwrotnym przypadku, jeśli chcesz otwarcie zadeklarować, że skalowalne obrazy znajdują się w danym katalogu, usuń z reguły wykrzyknik. Przykładowo, jeśli chciałbym, by moje obrazy znajdowały się w podfolderze *zwyciezca_nie_jest*, w pliku *.htaccess* należy wprowadzić taką oto zmianę:

```
<IfModule mod_rewrite.c>
  Options +FollowSymlinks RewriteEngine On
  # Adaptive-Images ---------------------------------------------------
  RewriteCond %{REQUEST_URI} zwyciezca_nie_jest
  # Wyślij żądanie o przesłanie dowolnego pliku GIF, JPG lub PNG, który
  ↪NIE ZOSTAŁ zachowany w wymienionych wyżej katalogach, do skryptu
  ↪adaptive-images.php, aby wybrał jego wersję mającą właściwy rozmiar.
  RewriteRule \.(?:jpe?g|gif|png)$ adaptive-images.php
  # END Adaptive-Images -----------------------------------------------
</IfModule>
```

Tyle wystarczy. To, czy narzędzie działa, sprawdzisz, umieszczając duży obraz na stronie i uruchamiając ją na smartfonie. Gdy otworzysz folder *ai-cache* w swoim ulubionym kliencie FTP, powinieneś zauważyć pliki i foldery z nazwami wziętymi od punktów kontrolnych transformacji Twojego układu, np. *480* (patrz zrzut poniżej).



Narzędzie Adaptive Images działa nie tylko w statycznych witrynach. Można go używać również w połączeniu z systemami zarządzania treścią; istnieje sposób na włączenie tego mechanizmu nawet w sytuacji, gdy przeglądarka użytkownika nie uruchomi kodu JavaScript. Korzystając z rozwiązania Adaptive Images, możesz przesłać różne grafiki w zależności od wielkości ekranu urządzenia klienta i zaoszczędzić na transferze danych w urządzeniach, które nie odnoszą korzyści z wczytywania domyślnych dużych obrazów.

Płynne siatki i zapytania medialne tworzą jedność

Jak pamiętasz, wcześniej w tym rozdziale poruszyliśmy problem rozchodzenia się elementów nawigacji pomiędzy wiele wierszy, gdy obszar operacyjny osiągnął pewną szerokość. Problem zostanie rozwiązany, jeżeli posłużymy się zapytaniem medialnym. Nasze linki rozpadają się

na szerokości 1060 pikseli i wyglądają znów dobrze w szerokości 768 pikseli (w miejscu, gdzie nasze wcześniej zdefiniowane zapytanie medialne weszło do akcji), więc czemu nie mielibyśmy zdefiniować dodatkowych stylów fontów w zakresie:

```
@media screen and (min-width: 1001px) and (max-width: 1080px) {
   #navigation ul li a { font-size: 1.4em; }
}
@media screen and (min-width: 805px) and (max-width: 1000px) {
   #navigation ul li a { font-size: 1.25em; }
}
@media screen and (min-width: 769px) and (max-width: 804px) {
   #navigation ul li a { font-size: 1.1em; }
}
```

Jak widać, zmieniamy wielkość fontu w zależności od szerokości obszaru operacyjnego i w rezultacie wszystkie elementy naszej nawigacji zawsze pozostają na swoim miejscu, od rozpiętości 769 pikseli po nieskończoność. Jest to oczywisty dowód silnej symbiozy między zapytaniami medialnymi a projektami płynnymi — zapytania medialne ograniczają niedostatki układów płynnych, a projekty płynne ułatwiają przejście z jednego układu witryny do innego.

System siatek CSS

Rozwiązania siatek CSS stanowią kontrowersyjny temat. Niektórzy projektanci je uwielbiają, inni ich nienawidzą. Nim zostanę zasypany górą emaili z pogróżkami, przyznam, że sam jestem niezdecydowany. Choć rozumiem deweloperów, którzy twierdzą, że są one zbędne i generują niepotrzebny kod, doceniam ich wartość, gdyż znacznie ułatwiają tworzenie prototypów układów.

Istnieje kilka rozwiązań generowania arkusza siatek CSS, które charakteryzują się różnym poziomem wsparcia dla idei RWD:

- Semantic (*http://semantic.gs*);
- Skeleton (*http://getskeleton.com*);
- Less Framework (*http://lessframework.com*);
- 1140 CSS Grid (*http://cssgrid.net*);
- Columnal (*http://www.columnal.com*).

Z tych wszystkich rozwiązań według mnie najlepszym jest system siatek Columnal, ponieważ generuje on płynną siatkę oraz zapytania medialne i wykorzystuje podobne zestawy klas do `960.gs`, bardzo popularnego stałego systemu siatek, który jest znany większości deweloperów.

Alpha, Omega i inne powszechnie spotykane klasy siatek
Wiele systemów siatek używa pewnych wspólnych nazw klas wykorzystywanych w definiowaniu układów. Łatwo odgadnąć funkcję klas row (rząd, szereg) i container (kontener, pojemnik), lecz istnieje wiele innych ich wariantów. Dlatego też warto zaznajomić się bliżej z dokumentacją systemu siatek — może to znacznie ułatwić Ci życie. Jednymi z najczęściej wykorzystywanych nazw klas w systemach siatek CSS są `alpha` i `omega` — opisują one odpowiednio pierwszy i ostatni element w szeregu (klasy `alpha` i `omega` usuwają wypełnienie i margines). Często spotyka się też klasę `col_x`, gdzie *x* oznacza liczbę kolumn układu, które dany element powinien zajmować (np. `col_6` oznacza 6 kolumn).

Błyskawiczne konstruowanie strony w systemie siatek

Załóżmy, że nie skonstruowaliśmy jeszcze płynnej siatki ani nie napisaliśmy żadnych zapytań medialnych. Dostaliśmy do rąk oryginalny plik PSD z projektem układu strony głównej *Zwycięzcą nie jest…* i powiedziano nam, że mamy jak najszybciej odwzorować jego strukturę w HTML i CSS. Sprawdźmy, czy system siatek Columnal pomoże nam sprostać temu wyzwaniu.

W naszym oryginalnym pliku PSD łatwo zauważymy, że układ opiera się na 16 kolumnach. Niestety system siatek Columnal obsługuje tylko 12 kolumn, więc spróbujmy nałożyć 12 kolumn na plik PSD, zamiast oryginalnych 16.

Po pobraniu archiwum ZIP i wypakowaniu jego zawartości stworzymy duplikat naszej witryny i w sekcji `<head>` strony odwołamy się do pliku *columnal.css* zamiast do *main.css*. Aby stworzyć wizualną strukturę przy użyciu układu Columnal, należy odnieść się do odpowiednich klas w kodzie HTML. Oto pełny kod naszej strony przed wprowadzeniem tych zmian:

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="viewport"  content="width=device-width,initial-scale=1.0" />
<title>Zwycięzcą nie jest…</title>
<script type="text/javascript">document.cookie='resolution='+Math.max
↪(screen.width,screen.height)+'; path=/';</script>
<link href="css/main.css" rel="stylesheet" type="text/css" />

</head>

<body>

<div id="wrapper">
   <!-- nagłówek i nawigacja -->
<div id="header">
      <div id="logo">Zwycięzcą <span>nie</span> jest...</div>
```


```
      <div id="navigation">
         <ul>
            <li><a href="#">Witryna</a></li>
            <li><a href="#">Streszczenie</a></li>
            <li><a href="#">Fotosy</a></li>
            <li><a href="#">Wideo</a></li>
            <li><a href="#">Cytaty</a></li>
            <li><a href="#">Quiz</a></li>
         </ul>
      </div>
   </div>
   <!-- treści -->
   <div id="content">
      <img class="oscarMain" src="img/oscar.png" alt="atwi_oscar" width="202"
      ↪height="531" />
      <h1>Rok w rok, <span>kiedy oglądam ceremonię rozdania Oscarów,
      ↪przepełnia mnie gorycz...</span></h1>
      <p>…na myśl, że takie filmy jak King Kong, Moulin Rouge czy Monachium
      ↪otrzymują statuetkę, a prawdziwe fenomeny kina przegrywają.
```

```
      Typowe dla Hollywood.</p>
<p>Pokażemy, które filmy są warte zachodu. </p>
   <a href="#">te filmy powinny wygrać &raquo;</a>
   </div>
   <!-- pasek boczny -->
   <div id="sidebar">
      <div class="sideBlock unSung">
         <h4>Zapomniani bohaterowie...</h4>
         <a href="#"><img src="img/midnightRun.jpg" alt="midnight_run"
         ↪width="99" height="135" /></a>
         <a href="#"><img src="img/wyattEarp.jpg" alt="wyatt-earp" width="99"
         ↪height="135" /></a>
      </div>
      <div class="sideBlock overHyped">
         <h4>Przereklamowane badziewie...</h4>
         <a href="#"><img src="img/moulinRouge.jpg" alt="MoulinRouge"
         ↪width="99" height="135" /></a>
         <a href="#"><img src="img/kingKong.jpg" alt="kingkong" width="99"
         ↪height="135" /></a>
      </div>
   </div>
   <!-- stopka -->
   <div id="footer">
      <p>Uwaga: liczy się tylko nasza opinia. Ty się mylisz, nawet jeśli wydaje
      ↪Ci się, że masz rację. Takie są fakty. Pogódź się z tym.</p>
   </div>

</div>
</body>
</html>
```

Na początek musimy potwierdzić, że nasz kontener `#wrapper` ma zawierać wszystkie inne elementy, więc dodajemy mu klasę `.container`:

```
<div id="wrapper" class="container">
```

Schodzimy trochę niżej, aż dotrzemy do frazy *ZWYCIĘZCĄ NIE JEST…* w pierwszym rzędzie. Dołączmy więc klasę `.row` do mieszczącego ją elementu:

```
<div id="header" class="row">
```

Nasze logo, choć ma tylko formę tekstową, mieści się w rzędzie, który wypełnia wszystkie 12 kolumn. Dlatego też dodajmy do jego komponentu deklarację `.col_12`:

```
<div id="logo" class="col_12">Zwycięzcą <span>nie</span> jest…</ div>
```

W następnym rzędzie widzimy nawigację — tutaj wstawmy klasę `.row`:

```
<div id="navigation" class="row">
```

Cały proces należy powtórzyć dla kolejnych elementów, dodając w odpowiednich miejscach klasy `.row` i `.col_x`. Przeskoczymy kawałek dalej, bo boję się, że uśniesz, gdy ja będę w kółko powtarzał ten proces. W zamian za to rzuć okiem na pełny kod po zmianach. Zauważ, że musieliśmy przesunąć obraz statuetki Oscara i umieścić go w innej kolumnie. Dodałem też dodatkowy element `.row` wokół komponentów `#content` i `#sidebar`.

```
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta name="viewport"  content="width=device-width,initial-scale=1.0" />
<title>Zwycięzcą nie jest…</title>
<script type="text/javascript">document.cookie='resolution='+Math.max
➥(screen.width,screen.height)+'; path=/';</script>
<link href="css/columnal.css" rel="stylesheet" type="text/css" />
<link href="css/custom.css" rel="stylesheet" type="text/css" />

</head>

<body>

<div id="wrapper" class="container">
   <!-- nagłówek i nawigacja  -->
   <div id="header" class="row">
      <div id="logo">Zwycięzcą <span>nie</span> jest...</div>
      <div id="navigation">
         <ul>
            <li><a href="#">Witryna</a></li>
            <li><a href="#">Streszczenie</a></li>
            <li><a href="#">Fotosy</a></li>
            <li><a href="#">Wideo</a></li>
            <li><a href="#">Cytaty</a></li>
            <li><a href="#">Quiz</a></li>
         </ul>
      </div>
   </div>
   <div class="row">

      <!-- treści  -->
      <div id="content" class="col_9 alpha omega">
         <img class="oscarMain col_3" src="img/oscar.png" alt="atwi_oscar" />
         <div class="col_6 omega">
         <h1>Rok w rok, <span>kiedy oglądam ceremonię rozdania Oscarów,
         ➥przepełnia mnie gorycz...</span></h1>
      <p>…na myśl, że takie filmy jak King Kong, Moulin Rouge czy Monachium
      ➥otrzymują statuetkę, a prawdziwe fenomeny kina przegrywają.
      Typowe dla Hollywood.</p>
```

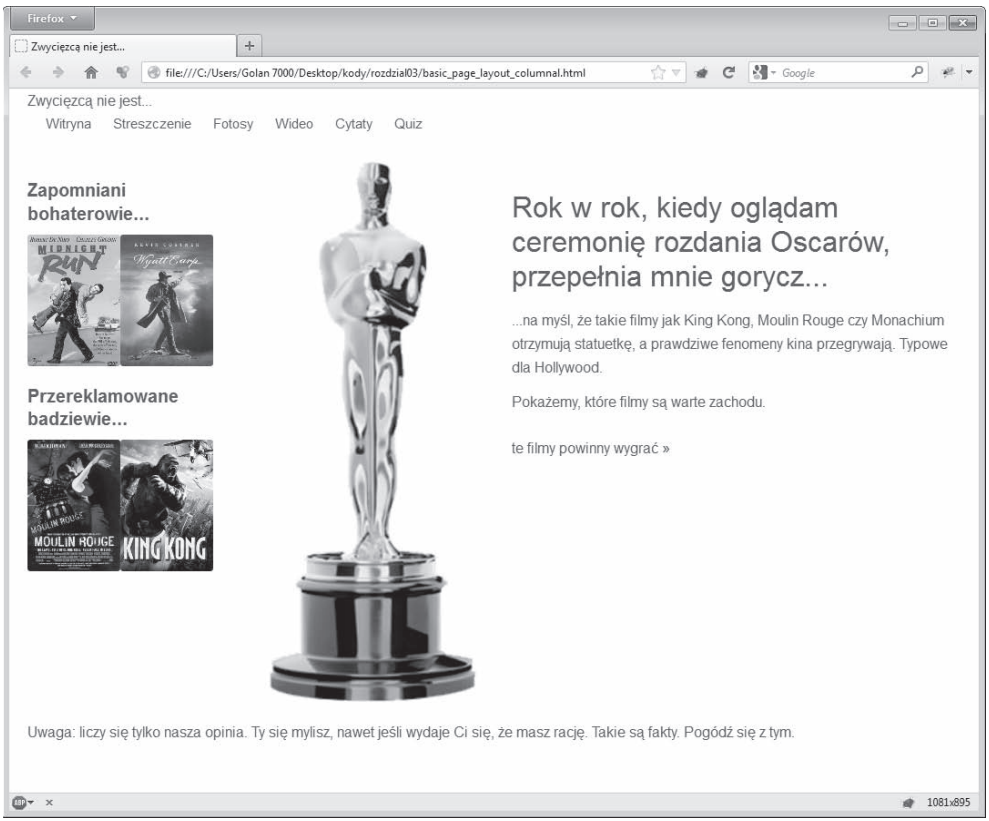
```
<p>Pokażemy, które filmy są warte zachodu. </p>
   <a href="#">te filmy powinny wygrać &raquo;</a>
      </div>
      </div>
      <!-- pasek boczny -->
      <div id="sidebar" class="col_3">
         <div class="sideBlock unSung">
            <h4>Zapomniani bohaterowie...</h4>
            <a href="#"><img src="img/midnightRun.jpg" alt="Midnight Run" /></a>
            <a href="#"><img class="sideImage" src="img/wyattEarp.jpg"
            ↪alt="Wyatt Earp" /></a>
         </div>
         <div class="sideBlock overHyped">
            <h4>Przereklamowane badziewie...</h4>
            <a href="#"><img src="img/moulinRouge.jpg" alt="Moulin Rouge" /></a>
            <a href="#"><img src="img/kingKong.jpg" alt="King Kong" /></a>
         </div>
      </div>
   </div>
   <!-- stopka -->
   <div id="footer" class="row">
      <p>Uwaga: liczy się tylko nasza opinia. Ty się mylisz, nawet jeśli wydaje
      ↪Ci się, że masz rację. Takie są fakty. Pogódź się z tym.</p>
   </div>

</div>
</body>
</html>
```

Konieczne okazało się też dodanie dodatkowych reguł CSS w pliku *custom.css*. Jego treść wygląda następująco:

```
#navigation ul li {
   display: inline-block;
}
#content {
   float: right;
}
#sidebar {
   float: left;
}
.sideBlock {
   width: 100%;
}
.sideBlock img {
   max-width: 45%;
   float:left;
}
.footer {
   float: left;
}
```

Po wprowadzeniu modyfikacji rzućmy okiem na naszą stronę, by upewnić się, że nasza struktura działa poprawnie i dopasowuje się do obszaru operacyjnego przeglądarki:



Jasne jest, że nad naszym układem trzeba będzie jeszcze trochę popracować (wiem, że jest to więcej, niż niedopowiedzenie), lecz jeśli musisz szybko stworzyć strukturę dokumentu, system siatek CSS w rodzaju Columnal jest wart Twojej uwagi.

Podsumowanie

W tym rozdziale dowiedziałeś się, jak można zmienić strukturę układu ze stałej, opartej na pikselach, na bardziej elastyczną, wyrażoną w procentach. Nauczyłeś się też używać jednostek `em` zamiast pikseli do definiowania wielkości kroju tekstu. Wiesz również, jak sprawić, by obrazy były skalowane w odpowiedzi na zmianę wielkości obszaru operacyjnego, oraz potrafisz używać serwerowego narzędzia służącego do wysyłania obrazów na urządzenie klienta w zależności od wymiarów jego ekranu. Eksperymentowaliśmy też z systemem siatek CSS zgodnym z filozofią RWD, który umożliwił nam stworzenie prototypu struktury strony przy minimalnym wysiłku.

Do tej pory na drodze ku udoskonaleniu umiejętności stosowania zasad RWD używaliśmy standardu HTML 4.01. W rozdziale 1., „Podstawy HTML5, CSS3 i projektowania elastycznych układów stron" wspomniałem o oszczędnościach, jakie wypływają z korzystania ze standardu HTML5. Oszczędności te są szczególnie ważne w przypadku układów elastycznych, które stawiają urządzenia mobilne na piedestale, ponieważ pozwalają na używanie przejrzystego, szybkiego we wczytywaniu i najbardziej semantycznego kodu z możliwych. W następnym rozdziale nauczysz się posługiwać się kodem HTML5 i wspólnie zmodyfikujemy kod strony, aby mogła wziąć to, co najlepsze, z najnowszej i najlepszej wersji tego standardu.

Skorowidz

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

T

U

W

Y

Z

Ż

Responsive Web Design
Projektowanie elastycznych witryn w HTML5 i CSS3

Jeżeli pamiętasz czasy przed rewolucją mobilną, to z łatwością przypomnisz sobie rozdzielczości królujące w tamtym okresie: 800×600 oraz 1024×768 pikseli. Nie było tabletów, smartfonów, netbooków i telewizorów podłączonych do internetu. Projektant stron WWW napotykał w swojej pracy wiele utrudnień, jednak nie musiał zmagać się z różnorodnością ekranów, jak webmasterzy naszej dekady. Obecnie projektowanie stron dopasowanych do wymogów różnych urządzeń to niezwykle cenna umiejętność.

Dzięki tej książce zdobędziesz ją w mig! Już za chwilę skorzystasz z możliwości HTML5 i CSS3, by stworzyć stronę, która zachwyci użytkownika, i to niezależnie od platformy, na której będzie jej używał. Co najważniejsze, żeby to osiągnąć, nie będziesz musiał przygotowywać kilku wersji jednej strony! W trakcie lektury poznasz nowości z CSS3 i HTML5. Dowiesz się, jak uzyskać rewelacyjne efekty, atrakcyjne wizualnie układy oraz decydować o rozmieszczeniu elementów w zależności od docelowej rozdzielczości. Książka ta jest obowiązkową lekturą dla każdego projektanta i programisty stron internetowych. Po prostu musisz ją mieć!

Przekonaj się, jak łatwo:

- obsługiwać różne rozdzielczości ekranów
- korzystać z plików multimedialnych
- wspierać starsze przeglądarki
- tworzyć lepsze strony WWW

Zaoszczędź czas — twórz uniwersalne strony WWW!

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

Informatyka w najlepszym wydaniu

sięgnij po WIĘCEJ

KOD KORZYŚCI

ISBN 978-83-246-6901-1

9 788324 669011

Cena: 49,00 zł