Przewodnik po tajnikach HTML5!

HTML5

nieoficjalny podręcznik

Wydanie II

Matthew MacDonald

O'REILLY

Helion

Tytuł oryginału: HTML5 The Missing Manual, 2nd Edition

Tłumaczenie: Maksymilian Gutowski
na podstawie: „HTML5. Nieoficjalny podręcznik” w tłumaczeniu Macieja Reszotnika

ISBN: 978-83-246-9251-4

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

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/htm5n2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/htm5n2.zip

Printed in Poland.

Spis treści

ROZDZIAŁ 7

Projektowanie elastycznych witryn w CSS3

Webdesignerzy musieli stawić czoło poważnemu wyzwaniu już wtedy, kiedy tylko zaczęli umieszczać materiały na stronach internetowych. Projektanci materiałów drukowanych mogli polegać na pewnych założeniach dotyczących wyglądu swoich dokumentów na papierze i sposobu, w jaki odbiorcy się nimi posługiwali, jednak w sieci panowało pod tym względem zupełne bezprawie. Ta sama strona HTML, w zależności od przeglądarek i preferencji użytkowników, mogła zostać wtłoczona do maleńkiego okienka albo mogła wisieć pośrodku ogromnego okna, zatem tworzenie rozbudowanych layoutów było ryzykowne. Layout, który wyglądał idealnie w jednym oknie, mógł się całkowicie rozłożyć w oknie o innych wymiarach.

Dziś to zróżnicowanie jest jeszcze większe. Webdeweloperzy muszą brać pod uwagę nie tylko różne wymiary przeglądarek na komputerach, ale także rozmiary różnych *urządzeń*, takich jak tablety i smartfony. Layouty jednocześnie zwiększyły swoją złożoność — dziś na większości witryn stykamy się z różnorakimi menu, pomocami nawigacyjnymi, paskami bocznymi i podobnymi elementami. Jeśli Twoim celem jest utworzenie jednej witryny, która ma wyglądać równie estetycznie w różnych warunkach, poradzenie sobie z takimi szczegółami będzie nie lada wyzwaniem.

Ponieważ webdesignerzy już dawno przenieśli kod odpowiedzialny za układ strony i formatowanie do plików CSS, można oczekiwać, że standard ten zapewnia jakieś rozwiązanie omówionego powyżej problemu. Rzeczywiście, CSS3 obsługuje **zapytania medialne**, które pozwalają stronie na przełączanie stylów w zależności od wielkości okna lub urządzenia, na którym jest wyświetlana.

Zapytania medialne są praktycznie nieodzowne w projektowaniu stron na urządzenia mobilne. Jeśli nawet nie przypuszczasz, że ktokolwiek będzie odwiedzać Twoje strony na smartfonie, zapytania dadzą pewność, że layout będzie się dostosowywał do środowiska pracy użytkownika, np. usuwając dodatkowe kolumny, gdy zabraknie na nie miejsca, albo przenosząc linki nawigacyjne z góry strony

na bok. Taka elastyczność wiąże się z popularną filozofią webdesignu, noszącą nazwę projektowania elastycznego (ang. *responsive design*), którą omówimy w tym rozdziale.

Podstawy projektowania elastycznego

Od początku istnienia sieci zróżnicowanie wymiarów okien było przyczyną wielu kłopotów. Z biegiem lat webdesignerzy opracowali szereg mniej lub bardziej eleganckich, komplementarnych technik, pozwalających na sprostanie wymogom projektowania elastycznego.

Przed zapoznaniem się z zapytaniami medialnymi warto przyjrzeć się ukazanym tutaj tradycyjnym metodom. Wszystkie są nadal ważne, ale — jak się przekonasz — nie stanowią pełnego rozwiązania. Znając ich ograniczenia, będziesz mógł zrozumieć, w jaki sposób CSS3 wypełnia wszystkie luki.

Płynny layout

Najprostszym sposobem zabezpieczenia strony przy zmianach wymiarów okna jest utworzenie layoutu **proporcjonalnego**, czyli takiego, który zajmuje całą dostępną powierzchnię, niezależnie od jej wielkości. Opracowanie proporcjonalnego layoutu jest — w teorii — całkiem proste. Podstawową zasadą jest podzielenie strony na kolumny o szerokościach określanych wartościami procentowymi, a nie pikselowymi. Powiedzmy, że Twoja strona ma następujący, dwukolumnowy układ:

```
<body>
  <div class="leftColumn">
    ...
  </div>

  <div class="rightColumn">
    ...
  </div>
</body>
```

Reguły stylów layoutu o stałych wymiarach mogą wyglądać następująco:

```
.leftColumn {
  width: 275px;
  float: left;
}

.rightColumn {
  width: 685px;
  float: left;
}

body {
  margin: 0px;
}
```

Tymczasem reguły stylów w zdefiniowanym proporcjonalnie układzie strony wyglądają tak:

```
.leftColumn {
  width: 28.6%;
  float: left;
}

.rightColumn {
  width: 71.4%;
  float: left;
}

body {
  margin: 0px;
}
```

Lewa kolumna ma tutaj szerokość `28.6%`, wobec czego zajmuje 28,6% szerokości swojego kontenera, elementu `<body>`. W tym przykładzie element `<body>` pozbawiony jest marginesów, a zatem rozciąga się na całą szerokość okna przeglądarki, przy czym lewa kolumna obejmuje 28,6% tego obszaru.

Jak można się domyślić, wartości procentowe obydwu kolumn wynoszą łącznie równo 100%. Kolumny rozciągają się i kurczą, aby dopasować się do szerokości okna przeglądarki. Layouty proporcjonalne nazywane są **płynnymi**, ponieważ zamieszczona w nich treść gładko dostosowuje się do dostępnej przestrzeni.

Uwaga: W tym przykładzie szerokość lewej kolumny (28,6%) wyliczono, dzieląc stałą szerokość kolumny (275 pikseli) przez szerokość całego układu (wynoszącą 960 pikseli, co jest częstym standardem). Ponieważ projektowanie większości układów zaczyna się zwykle na ustalonych szerokościach, webdeweloperzy są przyzwyczajeni do wykonywania takich obliczeń przy tworzeniu płynnych layoutów.

Rzecz jasna, dostosowanie szerokości samych kolumn nie wystarczy. Musisz również wziąć pod uwagę marginesy, odstępy i obramowania. Początkujący webdeweloperzy często wprowadzają w swoich pierwszych płynnych layoutach ustalone marginesy i odstępy (o wartościach pikselowych), a jednocześnie proporcjonalnie określają wymiary kolumn. Wskutek tego kolumny zajmują jedynie przestrzeń pozostałą po odjęciu szerokości marginesów od całej szerokości strony. Procentowe wartości szerokości kolumn są jednak wyliczane na podstawie całej szerokości strony, bez uwzględnienia marginesów. Ta rozbieżność może powodować problemy z wyświetlaniem strony w wąskich oknach, bo marginesy o wartościach pikselowych zabierają miejsce proporcjonalnie zdefiniowanym kolumnom.

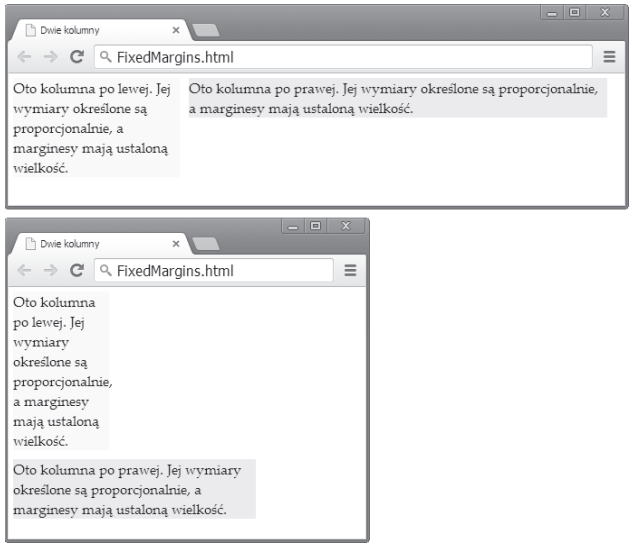
Zastanówmy się nad poniższymi stylami:

```
.leftColumn {
  width: 27%;
  margin: 5px;
  float: left;
}

.rightColumn {
  width: 68%;
  margin: 5px;
  float: left;
}
```

Te dwie kolumny zajmują łącznie 95% szerokości, pozostawiając dodatkowe 5% na marginesy. To wystarczy dla okien o średnich i dużych rozmiarach, ale w niewielkim oknie pozostałe 5% nie pomieści marginesów o ustalonej szerokości.

Aby zilustrować ten problem, nadaj obydwu kolumnom dwa różne kolory za pomocą własności `background`, a następnie pozmieniaj wymiary okna, tak jak na rysunku 7.1.



Rysunek 7.1. *Kolumny o proporcjonalnie określonej szerokości i marginesy o stałej szerokości nie łączą się dobrze. Kiedy okno jest za małe, marginesy wypychają drugą kolumnę z układu*

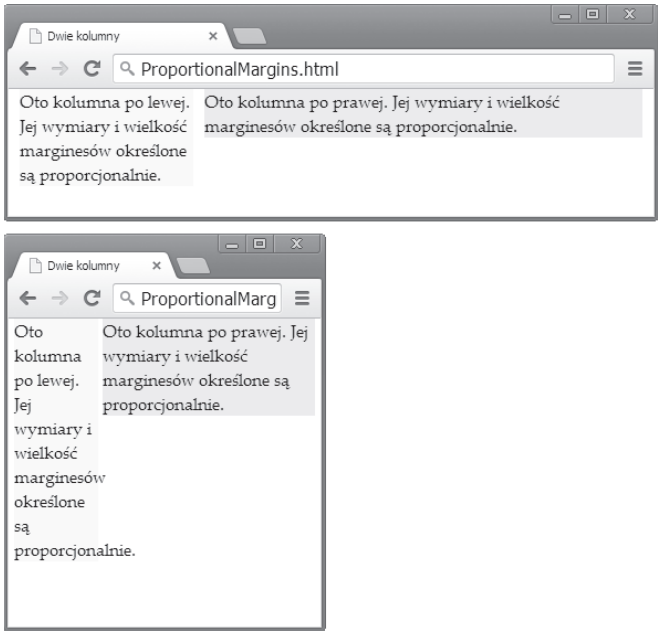
Rozwiązaniem tego problemu jest nadanie proporcjonalnej szerokości wszystkim marginesom, znajdującym się pomiędzy proporcjonalnymi kolumnami. Jeśli zatem kolumny pozostawiają 5% wolnego miejsca, możesz je przypisać marginesom. Podziel tę szerokość na trzy obszary o szerokości 1,66% — jeden na lewym krańcu okna, drugi na prawym i trzeci pomiędzy kolumnami.

```
.leftColumn {
  width: 27%;
  margin-left: 1.66%;
  margin-right: 1.66%;
  background: #FFFFCC;
  float: left;
}

.rightColumn {
  width: 68%;
  margin-right: 1.66%;
  background: #CCFFCC;
  float: left;
}
```

Na rysunku 7.2 widać efekt określenia szerokości zarówno kolumn, jak i marginesów proporcjonalnie przy użyciu wartości procentowych.

Proporcjonalne marginesy mogą Ci nie odpowiadać, jeśli potrzebujesz innego efektu. Jeżeli nie chcesz, by marginesy zmieniały wielkość stosownie do rozmiaru okna przeglądarki, możesz spróbować obejść to zachowanie. Możesz np. zamieścić dodatkowy element w jednej z proporcjonalnych kolumn i nadać mu własne marginesy lub odstępy o ustalonej szerokości. Ponieważ element ten umieszczony jest *w obrębie* zdefiniowanego już, w pełni proporcjonalnego układu, będzie się stale znajdował na swoim miejscu, niezależnie od wielkości okna.



Rysunek 7.2. *Layout strony będzie poprawnie skalowany do każdego rozmiaru, jeśli szerokości określone zostaną proporcjonalnie z wykorzystaniem wartości procentowych*

Obramowania sprawiają podobne problemy. Obramowanie nadane kolumnom zajmuje pewną ilość miejsca i może rozłożyć układ tak samo jak marginesy o ustalonej szerokości z rysunku 7.1. W takiej sytuacji problemu nie można rozwiązać poprzez wyliczenie proporcji, gdyż szerokości obramowania nie można określać w procentach. Najprostszym rozwiązaniem jest wykorzystanie techniki wspomnianej powyżej: osadź element `<div>` w kolumnie o procentowo określonej szerokości i nadaj mu obramowanie. Ta sprawdzona metoda wprawdzie nieco zaśmieca kod (zmuszając do utworzenia dodatkowej warstwy layoutu), ale dzięki temu układ strony może działać sprawnie w oknie o dowolnym rozmiarze.

Płynne obrazy

Wielokolumnowy, proporcjonalny układ strony jest podstawą elastycznego projektu. Kiedy jednak przychodzi czas na zajęcie się samą zawartością kolumn na stronie, pojawia się wiele innych kwestii, które trzeba wziąć pod uwagę.

Jedną z nich są obrazy. Pola obrazów standardowo dopasowują się do swojej zawartości, czyli dokładnie, co do piksela, przyjmują wymiary obrazu. Takie ustawienie może jednak wywoływać problemy w mniejszych oknach. Jeśli brakuje miejsca na wyświetlenie całego obrazu, wylewa się on z kolumny i rozchodzi na inne elementy, zakrywając je i wywołując niechlujne wrażenie.

Rozwiązanie tego problemu jest proste. Wystarczy nadać każdemu obrazowi maksymalną szerokość równą szerokości jego kontenera:

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

WIZJE PRZYSZŁOŚCI

Własność box-sizing i funkcja calc()

Omówione w tym punkcie problemy z layoutami są do tego stopnia powszechne, że w CSS3 można znaleźć mnóstwo potencjalnych rozwiązań. Oto dwa z najbardziej obiecujących, choć nie bez wad.

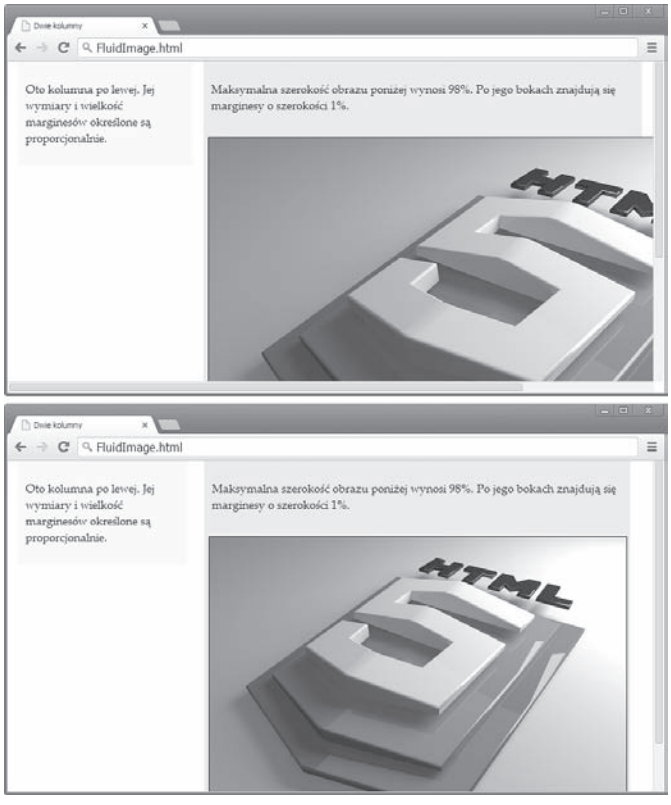
- **Box-sizing.** Obramowania standardowo zamieszczane są po zewnętrznej stronie elementów, przez co trzeba brać pod uwagę ich obszar w obliczeniach proporcji layoutu. W CSS3 pojawiła się jednak właściwość `box-sizing`, której wystarczy nadać wartość `border-box`, aby obramowanie znalazło się *w obrębie* pola elementu. Obramowanie wygląda wtedy tak samo, ale obliczenia wielkości takich komponentów ulegają zmianie. Oznacza to tyle, że kolumna szeroka na 67% zajmuje 67% szerokości kontenera, niezależnie od szerokości swojego obramowania.
- **Funkcja calc().** Jeśli musisz zestawić wymiary proporcjonalne z ustalonymi, przeprowadzenie stosownych obliczeń możesz powierzyć funkcji `calc()` w CSS3, a następnie wykorzystać ich wyniki w layoucie. Wyobraź sobie, że musisz utworzyć kolumnę szeroką na 67% kontenera bez 5 pikseli, które mają być przeznaczone na marginesy. Nieuważny webdeweloper może ominąć ten problem, zwężając kolumnę do 65%, co prowadzi do widocznej na rysunku 7.1 niespójności w rozmieszczeniu elementów. W CSS3 można jednak nadać własności `width` wartość `calc(67%-5px)`, dzięki której kolumna ma taką szerokość, jaką powinna — i ani piksela więcej.

W obu przypadkach wprowadzone rozwiązanie może jednak pogorszyć sytuację. Ustawienie `box-sizing` nie działa w IE 7, a w Firefoksie wymaga użycia przedrostka `-moz-` (jak wspomniano w rozdziale 6., w punkcie „Style właściwe konkretnym przeglądarkom"). Funkcja `calc()` nie działa w IE 7, IE 8 i przeglądarce Androida sprzed Chrome, a starsze wersje Safari wymagają podania przedrostka `-webkit-`. Problemom z obsługą można wprawdzie zaradzić za pomocą wypełnień, ale na razie łatwiej unikać korzystania z tych funkcji w ogóle, dopóki więcej ludzi nie przeniesie się na nowocześniejsze przeglądarki.

Jak zwykle, wartość 100% odnosi się do kontenera elementu. W tym przypadku jest to kolumna, w której zawarty jest obraz, a nie cała strona. Twój obraz może teraz osiągnąć *albo* swój maksymalny rozmiar, *albo* rozmiar ograniczony wielkością swojego kontenera, w zależności od tego, która z tych wartości jest mniejsza (rysunek 7.3).

Wskazówka: Jeśli zdecydujesz się utworzyć margines wokół obrazu, dopilnuj, by wartości procentowe własności `margin-left`, `margin-right`, oraz `max-width` sumowały się do 100% (a już na pewno nie przekraczały tej wartości).

Jednym ograniczeniem płynnych obrazów jest to, że przeglądarka musi pobrać obraz w pełnej wielkości, niezależnie od wymiarów, w jakich ma być wyświetlany. Wiąże się to z niewielkim, choć i tak zbędnym zużyciem czasu i transferu, co jest niedogodne dla użytkowników urządzeń mobilnych. Niestety, CSS nie poradzi sobie z tym problemem samodzielnie. Istnieją jednak pewne rozwiązania, w ramach których wykorzystuje się połączenie kodów działających po stronie serwera, serwisów internetowych oraz bibliotek JavaScriptu. Jeśli zamierzasz dostarczać użytkownikom urządzeń mobilnych bogatą w pliki graficzne stronę internetową, powinieneś rozważyć skorzystanie z technik, których omówienie znajdziesz w artykule Smashing Magazine pod adresem *http://tinyurl.com/responsive-img*. (Trzeba jednak przyznać, że wielu deweloperów w ogóle nie zawraca sobie tym głowy). Na szczęście, istnieje również sposób na rozwiązanie analogicznego, choć dużo poważniejszego problemu z wymiarami filmów na urządzeniach mobilnych (o którym przeczytasz w ramce na końcu tego rozdziału).



Rysunek 7.3. *Każdy obraz może wyrwać się spod kontroli, jeżeli nie określi się jego płynności (góra). Jeśli jednak ograniczysz jego szerokość do szerokości kontenera, obraz będzie wyświetlany w odpowiednich wymiarach (dół)*

Płynna typografia

Skoro utworzyłeś już płynny layout z obrazami o właściwie określonych wymiarach, czas zająć się tekstem w kolumnach. Niedoświadczeni webdeweloperzy często wybierają dla swojego tekstu adekwatny, ustalony rozmiar (określany wartościami pikselowymi) i na tym poprzestają. Takie sztywno określane wielkości nie przystają jednak do modelu skalowalnego projektowania układów stron, ponieważ tekst, który wygląda dobrze na ekranie komputera, jest niemal niewidoczny na urządzeniach mobilnych. Choć użytkownik zawsze może przybliżyć widok i przeczytać tekst, celem projektowania skalowalnego jest utworzenie strony pasującej do każdego okna, która nie wymaga częstego przybliżania i przewijania w poziomie.

I znowu rozwiązanie polega na unikaniu ustalonych jednostek miary w rodzaju pikseli i punktów. Zamiast tego wielkość tekstu należy ustalać względnie, korzystając z wartości procentowych i firetów. Najpopularniejszym wyborem jest firet, znany również jako jednostka **em**, której nazwa odnosi się do wielkiej litery M.

Uwaga: Firet od dawna służy do określania szerokości w typografii drukowanej. Pojęcie *em dash* (z ang. pauza) pierwotnie odnosiło się do poziomego znaku o szerokości równej wielkiej literze M danego kroju.

Wartości procentowe pozwalają na uzyskanie takiego samego wyniku: wielkość tekstu określana jest względem domyślnej wielkości tekstu przeglądarki. Jeśli wybierzesz rozmiar tekstu równy `110%` lub `1.1em`, otrzymasz pismo o 10% większe od zwyczajnego, nieobstylowanego tekstu. Wybierz rozmiar tekstu równy 50% lub 0.5em, aby zmniejszyć znaki o połowę.

Choć nie jest istotne, czy skorzystasz z wartości procentowych czy firetów, najbardziej elastyczne layouty bazują na jednej, określonej konwencji. W większości z nich wielkość bazowa tekstu na stronie wynosi 100% (wskazując, że jest to punkt odniesienia dla pozostałych rozmiarów), a wielkość tekstu innych elementów jest dodatkowo zwiększana lub zmniejszana firetami:

```
body {
  font-size: 100%
}

p {
  font-size: 0.9em
}

h1 {
  font-size: 2em
}
```

Doświadczeni webdeweloperzy nie poprzestają na tym, lecz używają firetów przy definiowaniu wszystkich innych wielkości w swoich layoutach. Jeśli np. gdzieś w głębi Twojego layoutu znajduje się drobny margines lub odstęp, lepiej zdefiniować go przy użyciu firetów, a nie pikseli. Takiego rodzaju wymiary są wtedy dopasowane do wielkości tekstu. Jest to dość subtelny czynnik, ale pozwala na dokładniejsze wycyzelowanie strony.

Wyobraź sobie np., że utworzyłeś dwupoziomowy layout, w którym element `<div>` osadzony jest w lewej kolumnie. Służy on do utworzenia dodatkowych odstępów dookoła treści bez przełamywania proporcjonalnego układu kolumn:

```
<body>
  <div class="leftColumn">
    <div class="leftColumnContent">
      ...
    </div>
  </div>

  <div class="rightColumn">
    ...
  </div>
</body>
```

Możesz określić obramowanie, marginesy i odstępy elementu `<div>` w lewej kolumnie przy użyciu wartości pikselowych. Taki layout jest sprawny i też powinien być płynny. Jednak lepiej użyć firetów, jak w poniższym przykładzie:

```
.leftColumn {
  width: 28.6%;
  background: #FFFFCC;
  float: left;
}

.rightColumn {
  width: 71.4%;
  background: #CCFFCC;
  float: left;
```

```
}

.leftColumnContent {
  border: 0.07em solid gray;
  margin: 0.3em;
  padding: 0.2em 0.3em 0.4em 0.4em;
}
```

Uwaga: Największą korzyścią z używania firetów do określania obramowań, marginesów i odstępów często jest to, że elementy te nie robią się za duże w małych oknach i nie dominują nad layoutem na urządzeniach mobilnych.

WIZJE PRZYSZŁOŚCI

CSS3: od em do rem

Gdy webdesignerzy używają tekstu w złożonych, skalowalnych layoutach, muszą się mierzyć z pewnym dziwactwem. Jednostki tekstowe o proporcjonalnych wartościach, takie jak firety i procenty, określają wielkość pisma w odniesieniu do kontenera. Nie sprawia to problemów w prostych przykładach ukazanych w punkcie „Płynna typografia", ponieważ kontenerem jest element `<body>`, w którym zawarta jest cała strona, bądź inny element, dziedziczący jego ustawienia fontu. Bólu głowy można dostać dopiero przy określaniu proporcjonalnych wielkości na *wielu* poziomach layoutu.

Załóżmy np., że utworzyłeś element `<div>` i nadałeś jego tekstowi wielkość `1.1em`. Następnie zamieszczasz w elemencie nagłówek `<h1>` o wielkości `2em`. Pewnie spodziewasz się, że tekst nagłówka będzie dwukrotnie większy od tekstu domyślnego, ale w rzeczywistości jest on dwukrotnie większy od tekstu *kontenera*, którego wielkość wynosi `1.1em`. Zatem nagłówek jest 2,2 razy większy od tekstu domyślnego.

Aby zapobiec wystąpieniu efektu kumulacji, musisz zachować rygor przy doborze miejsc, w których nadajesz tekstowi wielkość. Najlepiej robić to tylko na jednym poziomie layoutu. W CSS3 pojawiła się nowa jednostka, która pozwala na ominięcie tego problemu. Jest to `rem` (skrót od „root em", czyli `em` elementu głównego dokumentu). Jednostka `rem` jest w zasadzie taką samą jednostką względną jak `em`, z tym że — niezależnie od swojego położenia w arkuszu stylów — jej wartość zawsze jest obliczana w odniesieniu do wielkości tekstu elementu `<html>`, a nie kontenera. Wobec tego wartość `2rem` zawsze jest równa dwukrotności wielkości domyślnego tekstu, niezależnie od tego, gdzie się ją wprowadzi.

Jednostki `rem` są zdumiewająco dobrze obsługiwane — działają w każdej nowoczesnej przeglądarce. Problemy sprawiają jedynie dobrze znani maruderzy, czyli przeglądarki IE 8 i IE 7, które w ogóle nie rozumieją tych jednostek. Choć wypełnienie tej luki przy użyciu JavaScriptu jest możliwe (*http://tinyurl.com/rem-polyfill*), większość rozsądnych webdeweloperów woli unikać dodawania kolejnych skryptów, żeby tylko można było korzystać z nowych jednostek, i decyduje się dalej używać nieco niewygodnych jednostek `em`.

Oczywiście, typografia nie kończy się na definiowaniu wielkości tekstu. Aby utworzyć tekst, który pozostanie czytelny na różnorakich wyświetlaczach, musisz też wziąć pod uwagę szerokość wiersza, marginesy, interlinię, a nawet rozmieszczenie tekstu w kolumnach (co przedstawiono w punkcie „Wielokolumnowy tekst" rozdziału 6.). Żadnym z tych elementów nie można się zająć, jeśli używa się zwyczajnych, płynnych layoutów i określa wielkości wartościami proporcjonalnymi. *Możesz* jednak utworzyć bardziej elastyczne arkusze stylów, precyzujące takie szczegóły za pomocą zapytań medialnych. Wkrótce dowiesz się więcej na ten temat, ale najpierw trzeba rozpatrzyć kwestię automatycznego skalowania stron na telefonach komórkowych.

Widok strony: obsługa layoutu na smartfonach

Strona z dwiema kolumnami, którą omówiliśmy wcześniej, teoretycznie mogłaby się zmieścić w oknie o dowolnym rozmiarze. W praktyce jednak trzeba wziąć pod uwagę inne utrudnienie związane z korzystaniem z małych urządzeń mobilnych: rozmiar **widoku strony** (ang. *viewport*).

Apple wprowadził kiedyś widoki strony po to, żeby iPhone'y sensownie wyświetlały zwyczajne strony internetowe, które nie były oparte na technikach elastycznego projektowania. Zamiast wyświetlać niewielki fragment dużej strony internetowej, przeglądarki mobilne, takie jak Safari, pokazują całą stronę w pomniejszeniu, by uchwycić na ekranie więcej treści. Taki oddalony obszar wyświetlania określa się właśnie mianem widoku strony.

Technika ta opiera się na kompromisie. Z jednej strony sprawia, że strona wygląda tak, jak wyglądałaby na komputerze, ale przez to jednocześnie większość zwyczajnego tekstu staje się nieczytelna. Nie trzeba przewijać ekranu, ale trzeba częściej przybliżać i oddalać widok. Technika ta ułatwia poruszanie się po stronie, ale utrudnia odczytywanie treści.

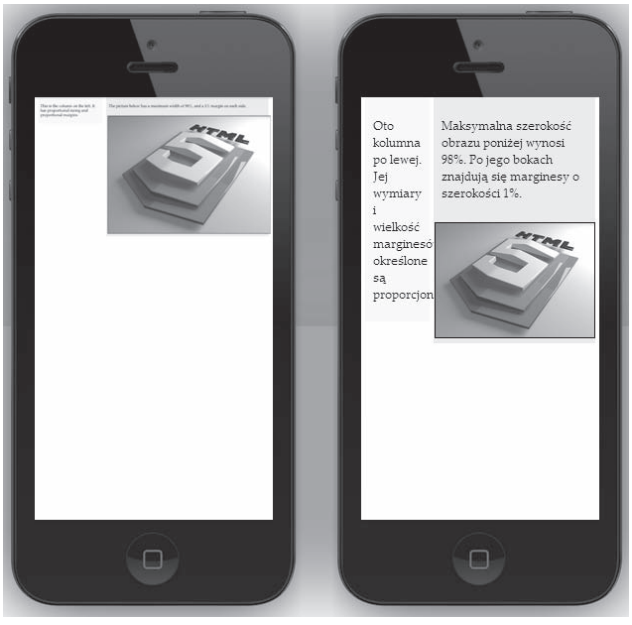
Uwaga: Choć widoki strony wprowadził Apple, obecnie korzystają z nich wszyscy inni producenci oprogramowania na urządzenia mobilne. Różnice dotyczą jedynie wielkości widoków i obszarów stron, jakie są wtłaczane w ich obręb.

Jeśli tworzysz tradycyjną witrynę na przeglądarki komputerowe, nie musisz się przejmować ustawieniami widoków strony urządzeń. One same dopilnują, by Twoja strona wyglądała w miarę dobrze na maleńkim ekranie telefonu (choć trzeba przyznać, że pomniejszona strona nie jest dla użytkowników szczególnie wygodna). Z drugiej strony, jeśli chcesz zaangażować się w projektowanie elastyczne i utworzyć witrynę przyjazną dla urządzeń mobilnych, musisz wprowadzić zmiany w opcjach widoków stron. Musisz polecić przeglądarkom mobilnym, by *nie skalowały* strony automatycznie. Można to wykonać, zamieszczając następujący element `<meta>` w sekcji `<head>` dokumentu:

```
<meta content="initial-scale=1.0" name="viewport">
```

Powyższy znacznik wskazuje urządzeniom mobilnym, by wyświetlały stronę w jej rzeczywistej skali, bez oddalania obrazu. Nowy iPhone umieści Twoją stronę w rzeczywistym rozmiarze w oknie o szerokości 320 pikseli. Bez takiego dostosowania skali iPhone przypisałby stronie 980 pikseli szerokości i wyświetlił ją w pomniejszeniu, by zmieściła się w oknie. Różnicę widać na rysunku 7.4.

Uwaga: Zapewne wiesz, że istnieje wiele symulatorów online, pozwalających na sprawdzenie wyglądu strony na różnych urządzeniach mobilnych. Na *http://mobiletest.com* można np. porównać wygląd strony na najnowszych iPhone'ach, iPadach i urządzeniach z Androidem. W większości z nich nie można jednak odtworzyć automatycznego skalowania. Innymi słowy, strona w symulatorze może działać tak, jakbyś przypisał wejściowej skali wartość `1.0` w ukazanym powyżej elemencie `<meta>`. Jeśli tego nie zrobiłeś, nie uzyskasz trafnego odwzorowania tego, co zobaczysz na samym urządzeniu, więc pozostań czujny na takie szczegóły.



Rysunek 7.4. *Po lewej: iPhone w ramach automatycznego skalowania traktuje widoczny płynny układ jak stronę zoptymalizowaną na potrzeby strony internetowej. Tekst jest nieczytelny bez przybliżenia obrazu.*

Po prawej: Po wyłączeniu skalowania strona wyświetlana jest w swojej właściwej postaci. Kolejnym krokiem jest zatem uproszczenie wyglądu layoutu na małych ekranach przy użyciu zapytań medialnych

Dostosowywanie layoutu przy użyciu zapytań medialnych

Wiesz już, jak utworzyć płynny layout, który rozszerza się i ścieśnia, aby dopasować się do okna przeglądarki o dowolnych wymiarach. Przyjmując takie podejście, zapewniasz, że strona będzie pasować do każdego okna. Nie zyskujesz jednak pewności, że w każdym oknie będzie wyglądać dobrze.

Skonstruowane w prosty sposób płynne layouty często psują się w skrajnych warunkach. W małym oknie liczne kolumny stają się żenująco wąskie, ścieśniając tekst i obrazy do nieczytelności. W bardzo dużym oknie kolumny stają się przygnębiająco szerokie — trudno nie zgubić się, próbując odczytywać rozciągnięte na pół strony wiersze.

Jednym sposobem na rozwiązanie tych problemów jest określenie ograniczeń rozciągania i ścieśniania layoutu. Wykonać to można przy użyciu własności `max-width` i `min-width`. Jeśli strona zostanie rozciągnięta ponad swój górny limit, po jej prawej pojawi się dodatkowy margines. Jeśli strona zostanie ścieśniona do rozmiaru mniejszego od dolnego limitu, kolumny zachowają swoją minimalną szerokość, a przeglądarka wyświetli paski przewijania. Ustawienia minimalnej i maksymalnej szerokości zapewniają podstawową ochronę przed wyświetlaniem ułomnych layoutów. Trzeba jednak przyznać, że jednocześnie zmniejszają elastyczność układu strony. Jeśli np. Twoja strona nie może zmniejszyć się do rozmiarów okna iPhone'a, nie będzie zbyt przydatna dla użytkowników korzystających z urządzeń mobilnych.

Lepszym rozwiązaniem jest subtelne zmodyfikowanie *struktury* layoutu przy zmianie wielkości strony. Do bardzo małego okna pasowałby przykładowo uproszczony layout bez pasków bocznych czy paneli z reklamami. Z kolei bardzo duże okno daje sposobność do powiększenia tekstu lub podzielenia go na wiele kolumn (punkt „Wielokolumnowy tekst" rozdziału 6.).

Odpowiedzią na to zapotrzebowanie są **zapytania medialne**. Ta własność CSS3 pozwala dynamicznie zmieniać formatowanie dla różnych urządzeń i ustawień wyświetlania. Użyta w odpowiedni sposób pozwoli Ci poprawnie wyświetlać strony na szerokich monitorach komputerów stacjonarnych i iPhone'ach — bez modyfikowania linijki kodu HTML.

Zapytania medialne

Zapytania medialne działają w oparciu o jedną z cech charakterystycznych urządzenia, które obsługują (takich jak wielkość ekranu, rozdzielczość, liczba wyświetlanych kolorów itd.). W oparciu o te dane możesz nałożyć różne style czy nawet inny ich arkusz.

Zapytanie medialne jest, najprościej rzecz ujmując, osobną sekcją arkusza stylów. Zaczyna się od terminu `@media`, po którym znajduje się warunek w nawiasie, a następnie seria powiązanych stylów w nawiasie klamrowym. Oto jego typowa struktura:

```
@media (media-feature-name: value) {
  /* Tutaj znajdują się nowe style. */
}
```

Zapytanie medialne przypomina instrukcję warunkową w JavaScripcie. Jeśli bieżąca przeglądarka spełnia warunki określone w nawiasie, zawarte w zapytaniu style stają się obowiązujące. Jeżeli jednak przeglądarka nie spełnia tych warunków, style są ignorowane.

Uwaga: Style znajdujące się poza sekcją `@media` nadawane są zawsze, niezależnie od okoliczności. Warunkowe style zapytania medialnego są nadawane jako *dodatek* do innych stylów. Z tego względu style warunkowe często muszą nadpisywać pozostałe ustawienia, np. ukrywając coś, co było poprzednio widoczne, przenosząc sekcję do nowego miejsca, nadając tekstowi nową wielkość itp.

Aby skorzystać z zapytania medialnego, musisz się zorientować, jakiego rodzaju warunki można utworzyć. Standard zapytań pozwala na sprawdzenie różnych szczegółów, zwanych **właściwościami mediów** (ang. *media features*). Możesz np. sprawdzić szerokość obszaru wyświetlania strony, a następnie podmienić style, gdy stanie się on mniejszy od jakiejś wartości. W tabeli 7.1 znajduje się lista najczęściej używanych właściwości mediów. (Istnieją też autorskie, eksperymentalne właściwości mediów, które nie są powszechnie obsługiwane i z tego względu nie znalazły się w tabeli).

Tabela 7.1. *Właściwości, które przydają się najbardziej przy tworzeniu zapytań medialnych*

Nazwa właściwości	Wartość	Zwykle służy do...

`width`
`min-width`
`max-width`
	Szerokość obszaru wyświetlania (lub drukowania).	zmiany layoutu, by dostosować go do bardzo wąskich (takich jak smartfon) albo bardzo szerokich wyświetlaczy.

`height`
`min-height`
`max-height`
	Wysokość obszaru wyświetlania.	zmiany layoutu, by dostosować go do bardzo wysokich lub krótkich wyświetlaczy.

`device-width`
`min-device-width`
`max-device-width`
	Pełna szerokość ekranu komputera lub innego urządzenia (bądź pełna szerokość zadrukowywanego arkusza).	dostosowania layoutu tak, by reagował na konkretne urządzenia, takie jak smartfony.

`device-height`
`min-device-height`
`max-device-height`
	Pełna wysokość ekranu lub arkusza.	dostosowania layoutu tak, by reagował na konkretne urządzenia, takie jak smartfony.

`orientation`	Jedna z dwóch wartości: `landscape` lub `portrait`.	zmiany layoutu na potrzeby różnych orientacji (poziomej i pionowej) ekranu.

`device-aspect-ratio`
`min-device-aspect--ratio`
`max-device-aspect--ratio`
	Proporcje obszaru wyświetlania, np. format obrazu 1/1 jest idealnym kwadratem.	dostosowania stylów, by pasowały do ekranów o różnych wymiarach (choć dość szybko wynikają z tego dalsze komplikacje).

`color`
`min-color`
`max-color`
	Liczba bitów koloru. 1-bitowa głębia koloru odpowiada obrazowi czarno-białemu, podczas gdy nowoczesne wyświetlacze standardowo mają 24-bitową głębię koloru, która przekłada się na miliony barw.	określania poziomu obsługi kolorów (np. w wersji strony internetowej do druku).

KĄCIK WSPOMNIEŃ

Typy mediów CSS

Ciekawe jest to, że już w edycji CSS 2.1 twórcy standardu poczynili pierwszą próbę rozwiązania problemu urządzeń mobilnych — wprowadzili tzw. typy mediów. Być może zdarzyło Ci się wcześniej z nich korzystać w celu dodania oddzielnego arkusza stylów dla druku stron.

```
<head>
  ...
  <!-- Używa tego arkusza do wyświetlania strony
       na ekranie -- >
  <link  rel="stylesheet" media="screen"
    href="styles.css">
  <!-- Używa tego arkusza do wydrukowania
       strony. -->
  <link rel="stylesheet" media="print"
    href="print_styles.css">
</head>
```

Atrybut `media` przyjmuje również wartość `handheld` — co dotyczy małoekranowych telefonów komórkowych z ograniczonym dostępem do Internetu. Większość mobilnych urządzeń zwraca uwagę na ten atrybut i używa odrębnego arkusza — jeśli istnieje. Niestety, atrybut `media` nie funkcjonuje do końca sprawnie i nie wystarcza, by poradzić sobie z wyborem i różnorodnością urządzeń, które obecnie istnieją. Metoda ta działa wciąż wyśmienicie, jeśli chodzi o wydruki.

Utworzenie prostego zapytania medialnego

Większość właściwości mediów występuje w kilku wersjach i pozwala na określenie górnych i dolnych limitów. Limity te są ważne, ponieważ większość zapytań medialnych odnosi się do sprecyzowanych zakresów wartości.

Przed wykorzystaniem zapytań medialnych wybierz własność, którą będziesz analizował. Jeśli np. chcesz utworzyć nowy zestaw stylów, który ma obowiązywać dla wąskich okien, powinieneś wybrać właściwość `max-width` i określić stosowny limit. W omawianym przykładzie nowy zestaw stylów wchodzi w życie, jeśli szerokość okna przeglądarki jest mniejsza niż 480 pikseli:

```
@media (max-width: 480px) {
  ...
}
```

Wskazówka: Aktualnie najpopularniejszymi zapytaniami są `max-device-width` (używane do formatowania stron na urządzenia przenośne), `max-width` (wykorzystywane do zmiany układu w zależności od rozmiaru okna) oraz `orientation` (stosowane w celu zmiany wyglądu strony w zależności od ułożenia smartfona w pionie lub poziomie).

Aby przeprowadzić prosty test, użyj zapytania medialnego do wprowadzenia widocznej zmiany. Poniższy kod zmienia kolor tła kolumny:

```
@media (max-width: 480px) {
  .leftColumn {
    background: lime;
  }
}
```

Sprawdź, czy zapytanie medialne działa. Zmniejsz powoli okno swojej przeglądarki. Gdy obszar wyświetlania stanie się węższy niż 480 pikseli, nowy styl uaktywni się, a kolumna zmieni barwę na limonkową. Wszystkie pozostałe reguły stylów określone dla klasy `leftColumn` (np. jej rozmiar i położenie) wciąż obowiązują, ponieważ zapytanie mediów ich nie nadpisuje.

Uwaga: Przeglądarki, które nie rozpoznają zapytań medialnych (np. Internet Explorer 8), po prostu zignorują nowe style i będą nakładać stare, niezależnie od tego, jak zmieni się wielkość okna.

Jeżeli chcesz, możesz dodać pomocniczą sekcję zapytań, która zniesie te nowsze reguły po jeszcze bardziej radykalnym zmniejszeniu wielkości okna. Przykładowo sekcja ta użyłaby nowego formatowania, gdy szerokość przeglądarki spadła poniżej 250 pikseli.

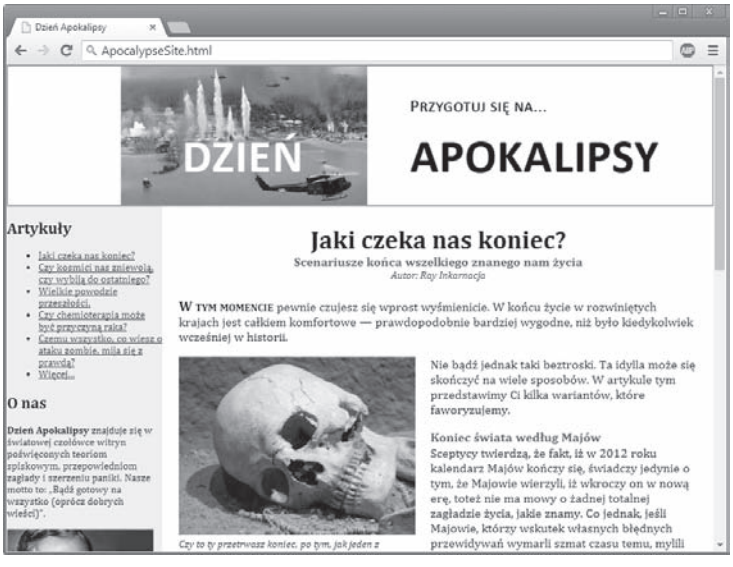
```
@media (max-width: 250px) {
  ...
}
```

Uważaj na to, że nowe style unieważniają wcześniej nałożone — czyli wszystkie własności ustawione w zwykłej deklaracji i w zapytaniu medialnym dla sekcji mniejszych niż 450 pikseli. Jeśli wydaje Ci się to zbyt skomplikowane, nie martw się. W dalszej części rozdziału dowiesz się, jak pracować z bardziej złożonymi medialnymi zapytaniami. Najpierw jednak zajmiemy się praktyczniejszym przykładem.

Layout przyjazny dla urządzeń mobilnych

Zapytania medialne są fundamentem witryny wyglądającej równie przyzwoicie w przeglądarce na komórce, jak w przeglądarce komputerowej. Wystarczy po nie zwyczajnie sięgnąć.

Na rysunku 7.5 widnieje podrasowana wersja strony *ApocalypseSite.html*, którą pamiętasz z rozdziału 2. Pierwotna strona oparta była na sztywnym layoucie o ustalonych szerokościach kolumn. Do utworzenia zmodyfikowanej wersji zastosowano wszystkie przedstawione w tym rozdziale techniki. Zbudowana jest na płynnym layoucie z elementami o proporcjonalnie określonych wymiarach (punkt „Płynny layout"), który dopasowuje się do każdej szerokości okna. Marginesy, odstępy, obramowania i wielkości tekstu określone są w nim firetami, dzięki czemu wszystkie te komponenty zachowują spójne wymiary na różnych urządzeniach (punkt „Płynna typografia"). Obraz nagłówka strony rozszerza się i kurczy, dopasowując się do dostępnego obszaru, a obraz reklamy w pasku bocznym, dzięki zastosowaniu techniki płynnych obrazów, nigdy nie wychodzi poza swoje pole (punkt „Płynne obrazy"). Użyto tu również metody z elementem `<meta>`, by zapobiec oddalaniu obrazu w przeglądarkach mobilnych (punkt „Widok strony: obsługa layoutu na smartfonach").



Rysunek 7.5. *Widoczna tutaj strona oparta jest na najlepszych technikach elastycznego projektowania. Leżący u jej podstaw kod CSS możesz pobrać ze strony helion.pl/ksiazki/htm5n2.htm*

Mówiąc krótko, nowa strona *ApocalypseSite.html* jest przystosowana do przeglądania na komórkach. Jej layout wciąż jednak nie jest *przyjazny* urządzeniom przenośnym. Jest tak, ponieważ bez względu na to, jak mocno dwie umieszczone obok siebie kolumny się ścieśnią, nie zmieszczą się schludnie w małym oknie. Musisz użyć zapytania medialnego, żeby zaradzić temu efektowi.

Zanim zaczniesz modyfikować swój arkusz stylów, musisz zastanowić się, jak mobilna wersja Twojej strony ma właściwie wyglądać. Strony na komórki zwykle zwężają się do pojedynczej kolumny. Paski boczne są wtedy albo ukrywane

zupełnie, albo przenoszone nad lub pod treść główną. Na rysunku 7.6 widnieje uporządkowana wersja *ApocalypseSite.html* na iPhone'a.

Rysunek 7.6. *Oto dwa zrzuty ekranowe mobilnej wersji strony ApocalypseSite.html. Strona zaczyna się od niewielkiego nagłówka, pod którym znajduje się treść artykułu (obrazek po lewej). Pasek boczny z linkami do artykułów i reklamą znajduje się po stopce artykułu (obrazek po prawej)*

Utworzenie mobilnej wersji *ApocalypseSite.html* jest zaskakująco proste. Zmniejszenie nagłówka witryny i dopasowanie tekstu następuje automatycznie za sprawą użycia płynnych obrazów i jednostek `em`. Pozostaje jedynie sprawić, by zapytanie medialne przestawiło kolumny.

Wygląd obydwu kolumn był pierwotnie zdefiniowany następującymi regułami:

```
.NavSidebar {
  float: left;
  width: 22%;
  font-size: small;
}
.Content {
  float: left;
  width: 78%;
}
```

Pasek boczny spływa na lewo i ma szerokość 22%. Pole z treścią główną znajduje się zaraz obok niego i ma szerokość 78%.

Ponieważ layout przestaje działać sprawnie na wąskich ekranach, warto skorzystać z popularnej właściwości mediów `max-width`. Jak dowiedziałeś się w poprzednim punkcie, `max-width` sprawdza bieżącą szerokość strony w oknie przeglądarki. Jeśli wartość ta jest niewielka, wyświetlanie dwóch kolumn jest nieodpowiednim rozwiązaniem.

Oto zapytanie medialne, za sprawą którego pola przestają pływać, a kolumny przyjmują całą dostępną szerokość okna:

```
@media (max-width: 568px) {
  .NavSidebar {
    float: none;
    width: auto;
  }

  .Content {
    float: none;
    width: auto;
  }
}
```

Powyższe style nadawane są jako dodatek do zdefiniowanych już stylów. Z tego względu konieczne może być przywrócenie zmodyfikowanym własnościom ich domyślnych wartości. W tym przykładzie style zapytań medialnych przywracają wartość `none` własności `float` i wartość `auto` własności `width` (choć równie dobrze można by jej nadać wartość `100%`). Są to wartości domyślne, ale podane pierwotnie reguły stylów paska bocznego zmodyfikowały je. Zauważ też, że pierwotne reguły stylów `NavSidebar` określały wielkość tekstu. Zapytanie medialne nie zmienia tej reguły, wobec czego ona nadal obowiązuje.

To zapytanie medialne, mówiąc ściśle, określa style odnoszące się do każdego wąskiego okna, niezależnie od tego, czy jest to okno przeglądarki na komórce, czy pomniejszone okno przeglądarki komputerowej. To zupełnie sensowne rozwiązanie, ale style można dodatkowo uszczegółowić i utworzyć osobne reguły dla maleńkich okien na komputerach oraz osobne dla urządzeń mobilnych. Warunki dla małych okien określa się właściwością `max-width`, a do wykrywania urządzeń mobilnych używa się właściwości `max-device-width` (o których wspomniano w tabeli 7.1).

Wskazówka: Określenie punktu, w którym następuje przełączenie na uproszczony layout, należy do Ciebie, ale warto zdecydować się na granicę 568 pikseli. Wynika to z tego, że 568 pikseli to szerokość strony na zorientowanym poziomo iPhonie. Dotyczy to również urządzeń działających na systemie Android (tabela 7.2).

W tym przykładzie trzeba wprowadzić jeszcze jedną poprawkę. W pierwotnej wersji strony sekcja `NavSidebar` zamieszczona była w kodzie HTML przed sekcją `Content`. Dzięki temu obydwie mogły spływać w lewo, a `NavSidebar` znajdował się przy lewej krawędzi strony. Po usunięciu przypisanych wartości `float` wersja mobilna witryny musi wyświetlać sekcje w takiej kolejności, w jakiej są umieszczone w pliku, przez co `NavSidebar` znajduje się na górze strony, a dopiero pod nim wyświetlana jest sekcja `Content`. Taki layout może zniechęcać użytkowników komórek, ponieważ przewijanie linków i reklam przed dotarciem do właściwej treści strony nie jest szczególnie wygodne.

Rozwiązaniem jest opracowanie kodu spełniającego wymogi mobilnej wersji strony, aby dopiero *potem* zdefiniować dodatkowe reguły CSS, tworzące bardziej rozbudowany, wielokolumnowy layout. Ta wzorcowa technika jest zgodna z projektowaniem **ukierunkowanym na urządzenia przenośne** (ang. *mobile-first*).

W omawianym przykładzie zastosowanie jej wiąże się z zamieszczeniem sekcji `Content` przed sekcją `NavSidebar`. Rozwiązuje to problem z mobilną wersją przeglądarki, ale jednocześnie w pełnej wersji strony spycha pasek boczny do prawej kolumny. Aby naprawić tę usterkę, postaraj się, by sekcja `Content` spływała w prawo:

```
.Content {
  float: right;
  width: 78%;
}
```

Sekcja `Content` powraca teraz na swoje miejsce po prawej, a `NavSidebar` trzyma się lewej strony, dzięki czemu przywrócony zostaje pełnowymiarowy layout z rysunku 7.5.

PORADNIK INŻYNIERA

Ukrywanie i zastępowanie sekcji

Jeśli jesteś szczególnie ambitny, możesz wprowadzić o wiele więcej zmian, aby odróżnić wersję strony na urządzenia mobilne od jej pełnego wydania. Możesz np. skorzystać z własności CSS `display` do ukrywania i wyświetlania całych *sekcji* strony.

Najpierw jednak musisz wziąć pod uwagę wady takiego rozwiązania. Jeśli chcesz podmienić duże fragmenty strony, prawdopodobnie uzyskasz rozbudowany kod, o którego utrzymanie, spójność i przetestowanie na różnych urządzeniach będziesz musiał później zadbać. Co więcej, przeglądarki pobierają obrazy zamieszczone w ukrytych sekcjach, choćby i miały ich nigdy nie wyświetlić. W przypadku urządzeń mobilnych takie zachowanie może poważnie obniżyć wydajność i niepotrzebnie obciążyć łącze.

Istnieje jednak sytuacja, w której podmiana sekcji *jest* wskazana: dzieje się tak, kiedy pojawia się konieczność zastąpienia rozbudowanego systemu nawigacyjnego skromniejszą, prostszą wersją na urządzenia przenośne. Dostarczanie użytkownikom mobilnym rozwijanych list nawigacyjnych zamiast nieporęcznych drzewek linków jest standardową praktyką. (Istnieje nawet sprytna technika, pozwalająca na przekształcenie rzędu odnośników w rozwijaną listę; znajdziesz ją na stronie *http://css-tricks.com/convert-menu-to-dropdown*).

Czasami proste sztuczki i drobne zmiany nie wystarczają. Konieczne może się okazać bardziej radykalne przekształcenie mobilnej wersji witryny. Masz do wyboru szereg rozwiązań o różnych stopniach złożoności i zaawansowania. Możesz utworzyć całkowicie osobną witrynę na urządzenia przenośne i umieścić ją w innej domenie („New York Times" tak właśnie robi ze swoją wersją na urządzenia mobilne *http://mobile.nytimes.com*). Wymaga to dużego nakładu pracy i uruchomienia na serwerze systemu zarządzania treścią, żeby zasoby dostępne na stronie mobilnej pokrywały się z zasobami standardowej witryny. Innym sposobem jest napisanie działającego po stronie serwera kodu, który sprawdza każde żądanie, identyfikuje przeglądarkę, z której użytkownik korzysta, i przesyła odpowiednie treści. To rozwiązanie jest idealne, jeśli dysponujesz odpowiednim czasem i umiejętnościami.

Skromniejsza metoda polega na zastosowaniu narzędzia JavaScriptu, pozwalającego dynamicznie modyfikować strony, w zależności od wyświetlacza. Jednym z nich jest Modernizr; za pomocą jego metody `Modernizr.mq()` można testować zawarte w kodzie zapytania medialne (*http://modernizr.com/docs*). Korzystanie z takiego narzędzia jest efektywniejsze od używania zapytań medialnych, ale prowadzi do zwiększenia złożoności układu strony.

Na tym etapie możesz rozważyć dodanie kolejnego zapytania medialnego, aby dostosować style także do bardzo szerokich okien, np. dzieląc tekst na wiele kolumn (przy użyciu własności CSS opisanych w rozdziale 6., w punkcie „Wielokolumnowy tekst") dla zachowania jego czytelności.

Wskazówka: Szukasz inspiracji? Skorzystaj z gotowych, elastycznych szablonów, których jest w sieci pełno. Zacznij od przejrzenia *http://html5up.net*, *www.typeandgrids.com* lub *http://responsify.it*.

Zapytania medialne — wyższa szkoła jazdy

Czasem do nałożenia konkretnych stylów powinno dojść po spełnieniu kilku odmiennych warunków. Taki scenariusz przedstawiono w poniższym przykładzie:

```
@media (min-width: 400px) and (max-width: 700px) {
   /* Nakłada formatowanie dla okien szerszych niż 400 pikseli i węższych od 700 pikseli. */
}
```

Takie zapytanie sprawdza się, gdy trzeba wprowadzić kilka zestawów wzajemnie wykluczających się stylów, lecz nie chce się mieć kłopotów z nakładaniem warstw reguł. Rzuć okiem na ten przykład:

```
/* Style uniwersalne */
@media (min-width: 600px) and (max-width: 700px) {
   /* Nakłada formatowanie dla okien szerszych niż 600 pikseli i węższych od 700 pikseli. */
}
@media (min-width: 400px) and (max-width: 599.99px) {
   /* Nakłada formatowanie dla okien szerszych niż 400 pikseli i węższych od 600 pikseli. */
}
@media (max-width: 399.99px) {
   /* Nakłada formatowanie dla okien węższych niż 400 pikseli. */
}
```

Od tej pory, gdy okno przeglądarki zmaleje do 380 pikseli, uruchomione zostaną dwa zestawy reguł: style standardowe i style z końcowego bloku `@media`. To, czy takie rozwiązanie upraszcza, czy komplikuje Ci życie, zależy od tego, co pragniesz osiągnąć. Jeżeli korzystasz z wielu stylów i często je zmieniasz, zaprezentowane tu podejście znacznie ułatwi pracę.

Pamiętaj jednak, że nawet tutaj musisz zwracać uwagę, które własności się nakładają. Jeżeli np. w dwóch miejscach ustawisz maksymalną i minimalną szerokość elementu na 400 pikseli, otrzymasz punkt, w którym oba style będą na siebie nachodzić. Trochę niezręcznym rozwiązaniem jest użycie ułamków, np. 399,99 pikseli, jak to miało miejsce w poprzednim przykładzie.

Innym pomysłem jest wykorzystanie słowa klucza `not`. Między obiema metodami nie ma żadnej widocznej różnicy, ale być może podejście w tym przykładzie będzie Ci lepiej odpowiadać:

```
/* Style standardowe */
@media (not max-width: 600px) and (max-width: 700px) {
   /* Nakłada formatowanie dla okien szerszych niż 600 pikseli i węższych od 700 pikseli. */
}
@media (not max-width: 400px) and (max-width: 600px) {
   /* Nakłada formatowanie dla okien szerszych niż 400 pikseli i węższych od 600 pikseli. */
}
@media (max-width: 400px) {
   /* Nakłada formatowanie dla okien węższych niż 400 pikseli. */
}
```

Istnieje jeszcze jeden sposób na zniesienie formatowania, którego nie podano w przeanalizowanych przykładach. Każdą sekcję zapytań medialnych (`@media`) otwierają standardowe, uniwersalne reguły. W zależności od wielu czynników, może się okazać, że lepiej będzie całkowicie rozdzielić segmenty arkusza (tak aby np. urządzenia mobilne korzystały z oddzielnego arkusza niezależnych stylów). W tym celu zapytań medialnych należy użyć w zewnętrznych arkuszach. Temu zagadnieniu poświęcono następny punkt.

Zastępowanie całego arkusza stylów

Jeśli w grę wchodzą jedynie drobne usprawnienia strony, blok `@media` okaże się przydatny w grupowaniu oddzielnych zestawów stylów w jednym pliku. Jeżeli jednak chcesz wprowadzić znaczne zmiany, może się okazać, że łatwiej będzie Ci zarządzać formatowaniem, kiedy utworzysz odrębny arkusz stylów. Odwołując się do niego, możesz zadeklarować zapytanie medialne.

```
<head>
  <link rel="stylesheet" href="standard.css">
  <link rel="stylesheet" media="(max-width: 568px)"
href="small_styles.css">
  ...
</head>
```

Przeglądarka ściągnie drugi arkusz (*small_styles.css*), lecz nie użyje go dopóty, dopóki szerokość jej okna nie spadnie poniżej maksymalnej wartości parametru `width`.

W poprzednim przykładzie nowe style usunęły starsze, już nałożone. W niektórych przypadkach lepiej utworzyć całkowicie oddzielne, niezależne arkusze. W tym celu dodaj do standardowego arkusza zapytanie medialne, upewniając się w ten sposób, że zostanie uruchomione tylko dla okien o dużych rozmiarach.

```
<link rel="stylesheet" media="(min-width: 568.01px)" href="standard.css">
<link rel="stylesheet" media="(max-width: 568px)"
href="small_styles.css">
```

Problem z tym rozwiązaniem polega na tym, że przeglądarki, które nie obsługują zapytań medialnych, zignorują *oba* arkusze. W starszych wersjach Internet Explorera usuniesz ten problem, dodając standardowy plik CSS raz jeszcze, wewnątrz specjalnego komentarza.

```
<link rel="stylesheet" media="(min-width: 568.01px)" href="standard.css">
<link rel="stylesheet" media="(max-width: 568px)"
href="small_styles.css">
<!--[if lt IE 9]>
  <link rel="stylesheet" href="standard.css">
<![endif]-->
```

Nie jest to rozwiązanie doskonałe. Starsze wersje Firefoksa (wcześniejsze niż 3.5) nie rozpoznają zapytań medialnych, komentarz warunkowy ich nie dotyczy. Problem ten da się rozwiązać po rozpoznaniu przeglądarki i „podmienieniu" strony przy użyciu kodu JavaScript, lecz nie jest to zbyt eleganckie wyjście. Na szczęście, starsze wersje Firefoksa są coraz rzadziej używane.

Notabene, możesz połączyć medialne zapytania z typami opisanymi w ramce „Typy mediów CSS". Zacznij od przypisania atrybutowi `media` typu, ale nie wpisuj go w nawiasach. Oto sposób na zadeklarowanie druku strony o ustalonej wielkości.

```
<link rel="stylesheet" media="print and (min-width: 25cm)"
 href="NormalPrintStyles.css" >
<link rel="stylesheet" media="print and (not min-width: 25cm)"
 href="NarrowPrintStyles.css" >
```

Rozpoznawanie urządzeń mobilnych

Jak już wiesz, komputery stacjonarne od urządzeń mobilnych odróżnisz, stosując zapytanie medialne z własnością `max-device-width`. Rodzi się pytanie, jakich szerokości w tym celu użyć.

Telefony komórkowe wykryjesz, sprawdzając, czy parametr `max-device-width` jest mniejszy niż 568 pikseli. Jest to najlepsza, najlepiej potwierdzona zasada. Reguła ta rozpoznaje iPhone'y i współczesne telefony działające w oparciu o system Android:

```
<link rel="stylesheet" media="(max-device-width: 568px)"
 href="mobile_styles.css">
```

Jeśli jesteś pasjonatem techniki, w Twoim mózgu powinna zapalić się czerwona lampka. W końcu, aktualna generacja urządzeń mobilnych używa małych ekranów o wysokiej rozdzielczości. Przykładowo iPhone 5 wyświetla siatkę 640×1136 pikseli na raz. Dlatego też może Ci się wydać, że dla niego własność `max-device-width` powinna być większa. Co zaskakujące, tak nie jest.

Rozważ jako przykład urządzenie iPhone 5. Informuje ono przeglądarkę, że szerokość jego ekranu wynosi 320 pikseli w orientacji pionowej, pomimo że jego ekran w rzeczywistości składa się z dwukrotnie większej liczby pikseli fizycznych. Dzięki temu strony internetowe nie przesyłają swoich pełnych, komputerowych wersji na wyświetlacze iPhone'a o szerokości 640 pikseli. Choć iPhone niewątpliwie może wyświetlić taką wersję strony, niewielki rozmiar jego pikseli uniemożliwiałby użytkownikowi jej odczytanie.

Większość urządzeń w dalszym ciągu tak się zachowuje. Dodają one współczynnik korekcji, tzw. **stosunek pikseli**. W iPhonie (w wersji 4. i nowszych) każdy piksel z perspektywy CSS jest równy dwóm pikselom na ekranie, a więc stosunek pikseli jest równy 2. W zasadzie łatwo utworzyć zapytanie medialne, które wykryje iPhone'a 4, ale zignoruje starsze wersje tego telefonu:

```
<link rel="stylesheet"
 media="(max-device-width: 480px) and (-webkit-min-device-pixel-ratio: 2)"
 href="iphone4.css">
```

W tabeli 7.2 widnieją szerokości niektórych z najpopularniejszych urządzeń. Pamiętaj, że często trzeba się liczyć z korekcją stosunku pikseli. Tak jest w przypadku iPada, którego wszystkie wersje zgłaszają szerokość 768 pikseli, pomimo że liczba pikseli fizycznych uległa podwojeniu w iPadzie 3.

Tabela 7.2. *Standardowe szerokości urządzeń*

Urządzenie	Szerokość urządzenia (w orientacji pionowej)	Szerokość urządzenia (w orientacji poziomej)
Apple iPhone 4	320	480
Apple iPhone 5	320	568
Apple iPad	768	1024
Samsung Galaxy S4	360	640
Google Nexus 4	384	640
Kindle Fire	600	1024

Wskazówka: Na rynek stale wychodzą nowe urządzenia. Aktualne informacje o nich znajdziesz na serwisach w rodzaju *www.mobitest.me/devices*).

Zidentyfikowanie iPada jest wyzwaniem innego typu. Użytkownicy mogą go obracać, co powoduje wyświetlenie treści w pionie lub w poziomie. Choć zmienia to parametr `max-width`, obrócenie urządzenia nie wpływa na własność `max-device-width`. W obu orientacjach — horyzontalnej i wertykalnej — iPad komunikuje, że jego szerokość wynosi 768 pikseli. Na szczęście, nie ma przeciwwskazań, by połączyć właściwości `max-device-width` i `orientation` — pozwoli to dynamicznie zmienić obstylowanie dokumentu w zależności od pozycji urządzenia.

```
<link rel="stylesheet"
 media="(max-device-width: 768px) and (orientation: portrait)"
 href="iPad_portrait.css">

<link rel="stylesheet"
 media="(max-device-width: 768px) and (orientation: landscape)"
 href="iPad_landscape.css">
```

Oczywiście, sposób ten sprawdza się nie tylko w iPadach. Gadżety wyposażone w wyświetlacz o podobnej szerokości (w tym przypadku 768 pikseli lub mniej) otrzymają ten sam zestaw reguł.

Uwaga: Zapytania medialne nie zmienią zwykłej strony w przyjazną dla użytkowników komórek. Należy wziąć pod uwagę takie czynniki jak wrażenia użytkownika. Dobrym pomysłem może okazać się rozbicie treści na mniejsze części (w rezultacie nie trzeba będzie tak często przewijać strony). Zaleca się również unikanie efektów i elementów interakcji, z których trudno skorzystać w interfejsie dotykowym (np. wyskakujących menu).

NIEOSZLIFOWANY DIAMENT

Zapytania medialne i wideo

Najbardziej oczywistą różnicą między witrynami dla komputerów osobistych a tworzonymi z myślą o urządzeniach przenośnych jest sposób wykorzystania nagrań wideo. Witryna dla telefonu komórkowego będzie wyświetlać plik wideo w mniejszym oknie, odczytując mniejszy plik. Powód takiego zachowania powinien być jasny — telefony komórkowe mają zarówno wolniejsze i bardziej kosztowne połączenie z siecią, jak i znacznie mniejszą moc obliczeniową potrzebną do odtworzenia nagrania.

Korzystając z omówionych wcześniej technik zapytań medialnych, możesz łatwo zmienić wielkość elementu `<video>`, tak by dopasował się do mobilnego ekranu. Podanie odrębnego linku do „odchudzonej" wersji nagrania wideo stanowi o wiele większy problem.

HTML5 podaje rozwiązanie: dołącza atrybut `media` bezpośrednio do elementu `<source>`. Jak dowiedziałeś się w rozdziale 5., element `<source>` służy do zadeklarowania pliku multimediów, który ma zostać odtworzony przez komponent `<video>`. Dodając atrybut `media`, możesz przypisać określony typ pliku do rodzaju urządzenia.

W niżej przedstawionym przykładzie plik *butterfly_mobile.mp4* jest przekazywany sprzętom wyposażonym w mały ekran. Inne urządzenia pobierają plik *butterfly.mp4* lub *butterfly.ogv*, w zależności od obsługiwanego formatu wideo.

```
<video controls width="400" height="300">
  <source src="butterfly_mobile.mp4"
   type="video/mp4"
   media="(max-device-width: 480px)">
  <source src="butterfly.mp4"
  ↪type="video/mp4">
  <source src="butterfly.ogv"
  ↪type="video/ogg">
</video>
```

Oczywiście, do Ciebie należy zakodowanie oddzielnej kopii nagrania dla użytkowników telefonów. Narzędzia do zapisywania multimediów są na ogół wyposażone w profile, które ułatwiają cały proces. Mogą np. zawierać opcje kodowania filmu w profilu „iPad video". W Twojej gestii leży też upewnienie się, że używasz formatu plików właściwego dla danego urządzenia (zwykle jest to format H.264), oraz przygotowanie kilku wariantów pliku dla każdej przeglądarki.

Skorowidz

B

C

D

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

X

Y

Z

ż

HTML5

nieoficjalny podręcznik

HTML5 na dobre zadomowił się w sieci. Z jego dobrodziejstw garściami czerpią projektanci witryn — i zaskakują użytkowników coraz bardziej pomysłowymi aplikacjami. HTML5 zapewnia zaawansowaną integrację ze środowiskiem przeglądarki internetowej, usługi geolokalizacyjne oraz doskonałe wsparcie dla materiałów multimedialnych czy aplikacji offline. Jeszcze całkiem niedawno nikt nie marzył o rzeczach, na jakie obecnie ten język programowania pozwala twórcom stron!

Książka, którą trzymasz w rękach, należy do cenionej serii **„Nieoficjalny podręcznik"**. Jej kolejne wydanie zostało poprawione, zaktualizowane i rozszerzone o mnóstwo nowych informacji. Dzięki tej publikacji nie musisz odkrywać tajników HTML5 na własną rękę. Liczne przykłady oraz szczegółowe omówienie najróżniejszych zagadnień sprawią, że w mig opanujesz niuanse tego języka. W trakcie lektury dowiesz się, jak wykorzystać nowy element canvas i jak bez trudu zlokalizować użytkownika, oraz zobaczysz, jak przechować kluczowe dane w bazie przeglądarki. Sięgnij po tę książkę i przekonaj się, jakie możliwości daje Ci HTML5!

HTML5 to:

- wsparcie dla plików multimedialnych
- usługi geolokalizacyjne
- wygodne przechowywanie danych
- aplikacje offline
- standard nowoczesnej sieci Internet

Poznaj możliwości HTML5 i zacznij korzystać z nich już dziś!

helion.pl
księgarnia internetowa

Nr katalogowy: 23314

Księgarnia internetowa
http://helion.pl

Zamówienia telefoniczne:
0 801 339900
0 601 339900

O'REILLY
Helion

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

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 978-83-246-9251-4

9 788324 692514

Cena: 77,00 zł