PROGRAMMER TO PROGRAMMER™

Wykorzystaj niezwykłe możliwości platform
WordPress, **Joomla!** oraz **Drupal**
i twórz zachwycające witryny dla urządzeń mobilnych!

Programowanie mobilnych stron internetowych z wykorzystaniem systemów CMS

James Pearce

Helion

Tytuł: Professional Mobile Web Development with WordPress®, Joomla!®, and Drupal®

Tłumaczenie: Tomasz Walczak

ISBN: 978-83-246-3610-5

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

Printed in Poland.

Spis treści

9

Projektowanie pod kątem urządzeń przenośnych

W TYM ROZDZIALE:

- Ważne podejścia związane z dostosowywaniem witryny internetowej i marki do odbiorców mobilnych
- Stosowane po stronie klienta techniki pomagające dostosować wygląd witryny do różnych rodzajów urządzeń przenośnych
- Sposoby wykorzystywania kodu z serwera (lub z chmury) do wykonywania podobnych zadań

W tym rozdziale poznasz trendy i techniki z obszaru projektowania witryn na różnorodne urządzenia przenośne. „Projektowanie" zwykle związane jest z wizualnymi odczuciami użytkowników i interakcją z urządzeniem, a nie z projektem architektury całej witryny (ten temat omówiono w innym miejscu).

Najważniejsze zagadnienie, które trzeba poruszyć na początku rozdziału, dotyczy tego, że projektowanie jest wysoce osobistą i twórczą czynnością. Każda witryna jest inna, a w procesie projektowania i implementowania „doświadczeń użytkowników" uczestniczą zarówno projektanci, jak i właściciele witryn oraz marek. Dotyczy to także urządzeń mobilnych, dlatego w tym rozdziale nie opisano jednego sposobu projektowania witryn, a jedynie podano rodzaje zagadnień, które należy omówić i rozważyć w czasie rozwijania eleganckich serwisów mobilnych. Opisywanie estetycznych aspektów grafiki i projektowania interfejsu użytkownika zanadto wykracza poza główny temat tej książki.

W czasie, gdy powstaje ta książka, można czasem odnieść wrażenie, że w internetowych interfejsach użytkownika w urządzeniach mobilnych pojawia się zaskakująco mało nowatorskich pomysłów. Wiele aplikacji i witryn mobilnych najwyraźniej opartych jest na założeniu, że należy odzwierciedlać pewne standardowe wytyczne (zwłaszcza udostępniane przez producentów urządzeń i przeglądarek), czego efektem jest irytująca jednorodność. Żadne dwie witryny stacjonarne nie wyglądają tak samo, dlaczego więc ograniczać się do tworzenia witryn mobilnych podobnych do innych serwisów?

W tym rozdziale poznasz wiele wytycznych i narzędzi. Zobaczysz przede wszystkim, jak zapewnić dostosowanie projektu do różnych urządzeń (zwłaszcza po stronie klienta). Postaraj się jednak wykorzystać to tylko jako punkt wyjścia. Era tworzenia innowacyjnych projektów witryn mobilnych musi się dopiero rozpocząć, dlatego bądź tak twórczy, jak potrafisz!

Każda osoba projektująca interfejs mobilny szybko zauważy, że jednym z największych problemów jest utworzenie interfejsu, który wygląda świetnie w wielu różnych systemach. Możliwość rozwijania witryn oglądanych na urządzeniach przenośnych tylko jednego typu jest rzadkim przywilejem (tworzenie rozwiązań na jedno urządzenie nie jest dobrym pomysłem), dlatego każdy programista i projektant takich serwisów musi znać wiele sztuczek oraz technik radzenia sobie z różnorodnością urządzeń. W tym rozdziale zobaczysz niektóre ważne nowoczesne pomysły pozwalające rozwiązać ten problem.

Standardowe podejścia

W tym podrozdziale poznasz wybrane ogólne pomysły z obszaru projektowania interfejsów użytkownika. Mam nadzieję, że uznasz, iż wiele technik sprowadza się do uwzględniania zdrowego rozsądku, jednak w procesie rozwijania witryn w konkretnym systemie CMS posłużą one jako przydatne wskazówki.

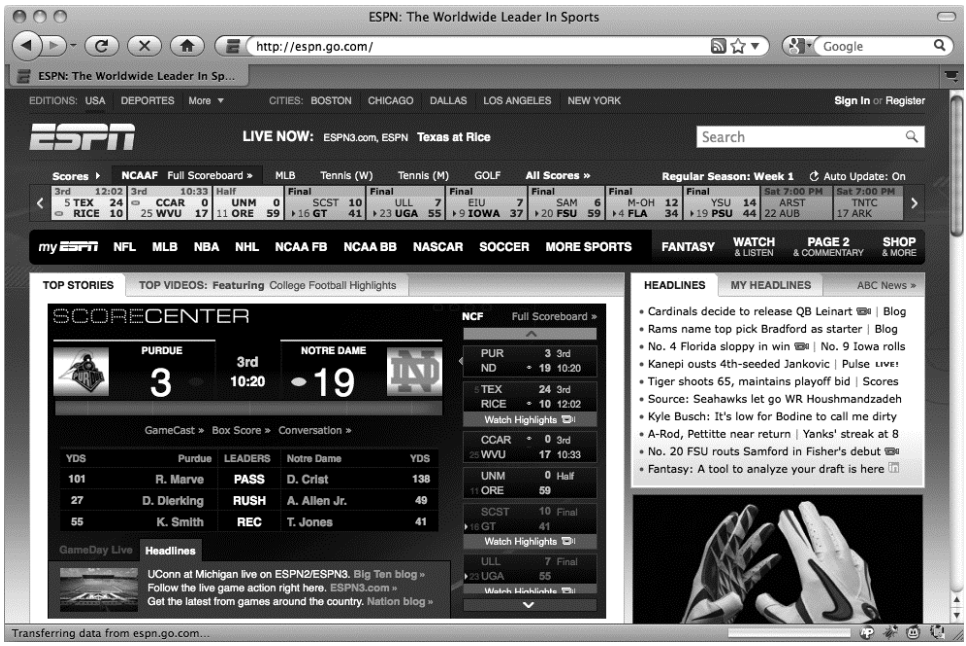
Zachowanie wizerunku marki

Jeśli Twoja obecność w internecie ogranicza się do prywatnego bloga z rzadko pisanymi artykułami, „marka" może obejmować tylko Ciebie, nazwę witryny i domyślny motyw wybrany spośród dostępnych w systemie CMS. Jeżeli jednak jesteś poważnym blogerem, możliwe, że korzystasz z zaawansowanego lub niestandardowego motywu. Ponadto jeśli firma korzysta z systemu CMS do prowadzenia witryny, nazwę, wygląd, styl i ogólny wizerunek marki przekazywany poprzez witrynę należy starannie dostosować do działalności pozainternetowej.

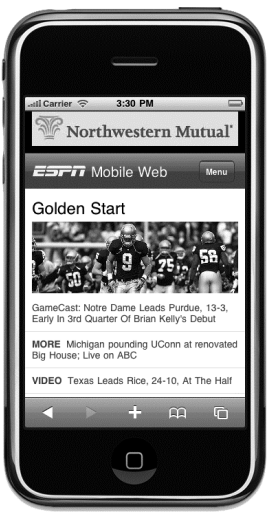
Dlaczego sytuacja miałaby wyglądać inaczej w witrynie mobilnej? Jeśli prowadzisz prosty blog, możesz zastosować standardowy motyw mobilny udostępniany wraz z wtyczką mobilną lub pobrany z sieci WWW. Jednak przy witrynach innego rodzaju urządzenia przenośne są bardzo ważnym kanałem kontaktu z czytelnikami, klientami lub partnerami biznesowymi — czasem nawet istotniejszym niż witryna stacjonarna! Niezwykle ważne jest, aby wygląd stron i wizerunek marki w witrynie mobilnej były zgodne przynajmniej z witryną stacjonarną, a najlepiej z wszystkimi materiałami dostępnymi w internecie i poza nim.

Jeśli poprawnie wykonasz zadanie, przełączanie się między różnymi wersjami serwisów będzie dla użytkowników niezauważalne i zrozumiałe. Przyjrzyj się witrynom sieci ESPN (rysunki 9.1, 9.2 i 9.3). Choć nie wszystkie aspekty (a nawet materiały) witryny stacjonarnej zostały starannie odzwierciedlone w wersjach mobilnych, nie ma wątpliwości, że każda wersja należy do tej samej firmy. Logo, schemat kolorów i wyróżnione informacje są w mniejszym lub większym stopniu spójne, choć układ w każdej wersji jest inny.

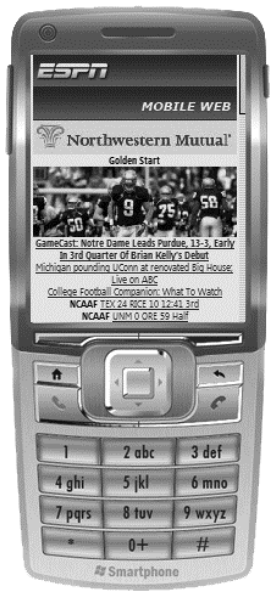
Warto porównać te serwisy z blogiem TechCrunch z informacjami z dziedziny technologii. Wersja stacjonarna (rysunek 9.4) rozpoczyna się od paska funkcji w górnej części strony, a dalej znajduje się lista artykułów. W nagłówku widoczne jest duże i rozpoznawalne logo.



Rysunek 9.1. Wersja stacjonarna

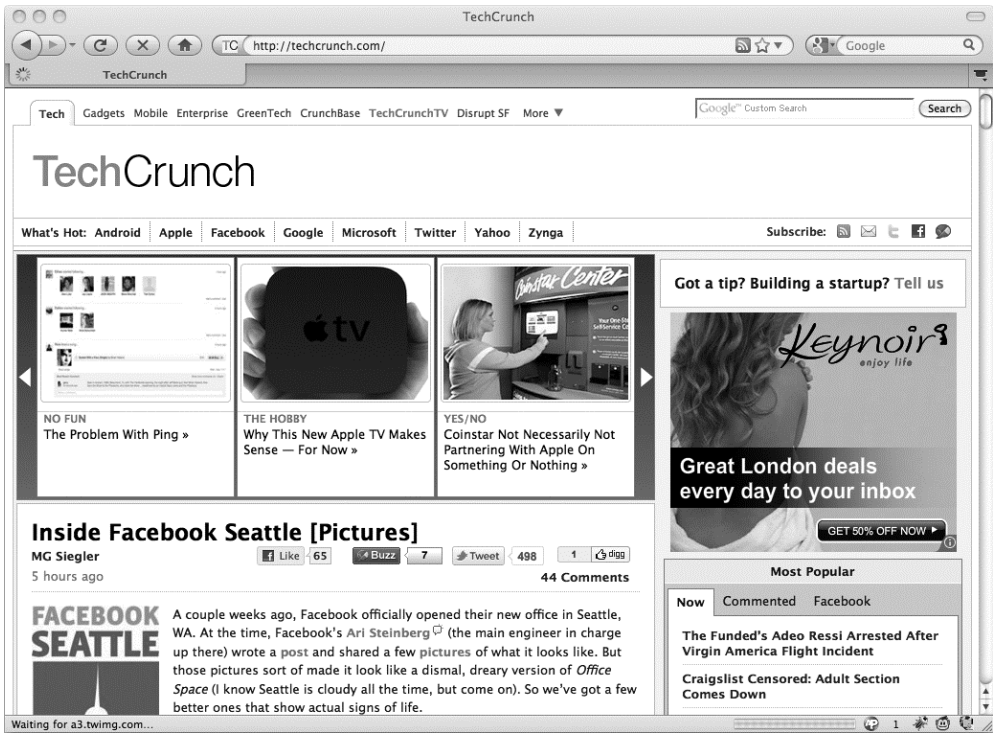


Rysunek 9.2. Wersja mobilna w iPhone'ie



Rysunek 9.3. Wersja mobilna w systemie Windows Mobile

Dziwne jest to, że w czasie powstawania tej książki ani witryna dla urządzeń z wyświetlaczem dotykowym (rysunek 9.5), ani witryna wyświetlana w starszych urządzeniach (rysunek 9.6) nie przypominały wersji stacjonarnej.



Rysunek 9.4. Wersja stacjonarna bloga TechCrunch

Rysunek 9.5. Blog TechCrunch na urządzeniu z wyświetlaczem dotykowym

Rysunek 9.6. Blog TechCrunch na starszym urządzeniu

Dla właściciela witryny lub programisty jest to niewybaczalny błąd, oznaczający utratę okazji do utworzenia spójnego wizerunku marki i jednolitych doświadczeń użytkowników w wielu kanałach.

Należy zapewnić przynajmniej spójność kolorów i zastosować logo z wersji stacjonarnej. Niewłaściwe jest też tworzenie witryn na podstawie natywnego menu iPhone'a (zwłaszcza na urządzenia z systemem Windows Mobile).

Także użytkownik może być zdezorientowany. Czy nadal ogląda witrynę TechCrunch? Dlaczego nie wygląda ona jak wersja stacjonarna? Gdzie jest logo? A może to witryna oszustów podszywających się za twórców serwisu? Dlaczego witryna przypomina menu w iPhone'ie, choć wiem, że korzystam z przeglądarki? Dlaczego witryna wygląda dokładnie tak jak serwis *NFL.com* (rysunek 9.7) i tysiące innych witryn?

Rysunek 9.7. Wersja mobilna przypomina witrynę NFL.com i wiele innych serwisów

Czy wynika to z tego, że właściciele wszystkich tych witryn zastosowali domyślny motyw z systemu CMS i nie zastanowili się nad zachowaniem wizerunku marki? Tak, jednak można uznać to za przypadkowy błąd. Internet mobilny jest nowym i nieznanym medium, dlatego brakuje dojrzałości w zakresie sposobów projektowania i rozwijania witryn mobilnych (i prawdopodobnie także umiejętności zapewniania spójnego wyglądu oraz tworzenia projektów równie dobrych jak w stacjonarnych odpowiednikach).

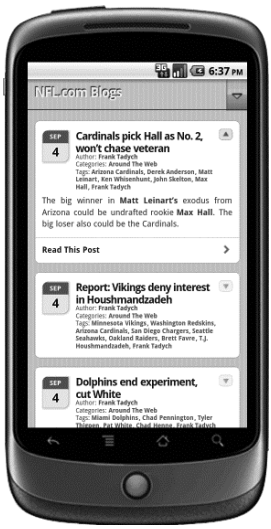
Dobra wiadomość jest taka, że zapewnienie spójnego wyglądu wersji mobilnej i stacjonarnej nie jest specjalnie trudne. Nawet dopasowanie schematu kolorów i zastosowanie wyraźnych elementów wizerunku firmy (na przykład kolorów dodatkowych) robi wielką różnicę, przy czym trzeba zachować ostrożność, aby przy wybranych ustawieniach tekst w wersji mobilnej był wyraźny, czytelny i kontrastowy. Powierzchnia ekranu nie jest duża, dlatego nie trzeba odtwarzać wszystkich dekoracji i aspektów wersji stacjonarnej.

Wykorzystanie natywnych wzorców projektowych

Wraz z pojawieniem się w 2007 roku iPhone'a firmy Apple świat mobilnych interfejsów użytkownika poszedł znacznie do przodu. W iPhone'ie wprowadzono wiele przełomowych zmian w samym urządzeniu, mobilnym systemie operacyjnym i przeglądarce, a wszystko to uzupełniał elegancki i spójny interfejs, dzięki któremu korzystanie z urządzenia było niezwykle proste. iPhone oznaczał początek powszechnego dostępu do internetu mobilnego oraz stanowił punkt odniesienia dla twórców mobilnych interfejsów użytkownika.

Niestety, to, że urządzenie stało się wzorcem, doprowadziło do nieoczekiwanych efektów ubocznych. Ponieważ interfejs użytkownika systemu operacyjnego został tak dobrze przyjęty, początkujący programiści rozwiązań dla internetu mobilnego wykorzystali go jako punkt wyjścia do rozwijania własnych projektów. Sama firma Apple przez udostępnienie wartego przeczytania dokumentu „Human Interface Guidelines for Web Applications" ułatwiła twórcom witryn stosowanie tych samych paradygmatów i wzorców, co w natywnych aplikacjach i samym urządzeniu.

Z jednej strony jest to doskonały punkt wyjścia. Przez zastosowanie technik interfejsu użytkownika znanych odbiorcom z innych części urządzenia można zagwarantować, że korzystanie z witryny będzie łatwe i naturalne. Jednak z drugiej strony łatwo można doprowadzić do skrajności — do sytuacji, w której witryna stanie się *nieodróżnialna* od aplikacji natywnych. Co gorsza, ślepe odzwierciedlanie systemu operacyjnego oznacza, że witryna wygląda na dobrze zintegrowaną i spójną w jednym konkretnym urządzeniu, ale zupełnie nieodpowiednio we wszystkich pozostałych. Jak pokazano w poprzednim podrozdziale, akceptowalne (choć niespójne z wizerunkiem marki) jest stosowanie pasków i zaokrąglonych rogów w stylu iPhone'a na blogu, jeśli jest wyświetlany w iPhone'ie, jednak ten sam motyw w Androidzie jest zupełnie nieadekwatny i wygląda dziwnie, co pokazano na rysunku 9.8 na przykładzie wcześniejszej witryny NFL.



Rysunek 9.8. Interfejs w stylu iPhone'a w systemie Android nie wygląda dobrze

Wyraźnie widać, że złotym środkiem jest połączenie obu podejść. Podstawowe reguły działania interfejsu użytkownika w witrynie nie powinny być obce lub nieoczekiwane dla osoby, która codziennie korzysta z urządzenia do odwiedzania innych witryn. Oczywiste jest też, że witryna nie powinna zakłócać pracy specyficznych dla urządzeń elementów interfejsu użytkownika, na przykład wyjątkowego sposobu wyświetlania znaczników `<select>` w iPhone'ie, co pokazano na rysunku 9.9.

Jednak naiwne odtwarzanie systemu operacyjnego konkretnego urządzenia (w skrajnym stopniu co do piksela) jest prawdopodobnie błędem. Takie podejście zaskakuje użytkowników innych urządzeń, ogranicza kreatywność i, co ważniejsze, uniemożliwia zapewnienie spójnego wizerunku marki względem witryny stacjonarnej.

Warto stosować współczesne i standardowe operacje oraz wzorce interfejsów użytkownika, jednak nie należy robić tego kosztem ograniczania własnych pomysłów z zakresu kosmetyki i estetyki witryny. Przyjrzyj się ponownie witrynie ESPN na iPhone'ie (rysunek 9.2). Autorzy witryny byli wyraźnie

Rysunek 9.9. Specyficzna obsługa znacznika <select> w iPhone'ie

zainspirowani rozwiązaniami z iPhone'a (takimi jak rozwijane menu i przyciski na pasku narzędzi), jednak zbudowali ją w taki sposób, że możliwe było zachowanie marki ESPN i indywidualnego charakteru witryny.

Witryna mobilna jako punkt wyjścia

W społeczności projektantów rozwiązań dla sieci WWW i internetu mobilnego coraz więcej zwolenników zyskuje nowe podejście do projektowania materiałów internetowych. Zgodnie z nim należy zacząć od mobilnej wersji witryny lub aplikacji, a następnie opracować wersję stacjonarną. Podejście to jest atrakcyjne z kilku powodów i zwłaszcza dla twórców witryn lub serwisów, które nie mają istniejącej wersji stacjonarnej. Przykładowo: zwolennicy omawianej techniki podkreślają znaczny wzrost ruchu w internecie mobilnym w ostatnich kilku latach, co oznacza, że niedługo wersja mobilna może generować istotną część łącznego ruchu w witrynach firmy.

Ponadto urządzenia przenośne udostępniają wiele możliwości nieobecnych w urządzeniach stacjonarnych. Zagadnienie to omówiono w rozdziale 5. W urządzeniach z obsługą geolokalizacji, wykrywaniem orientacji, ekranem dotykowym i aparatem funkcje witryny mobilnej mogą być *bogatsze* niż wersji stacjonarnej. W pewnym sensie lepiej jest zbudować wersję mobilną, a później ograniczyć jej funkcje pod kątem użytkowników stacjonarnych. Ta kwestia stanie się ważniejsza w przyszłości, kiedy więcej możliwości urządzenia będzie dostępnych poprzez standardowe interfejsy API dla aplikacji działających w przeglądarkach.

Na zakończenie warto uwzględnić odmienne podejście — trzeba umożliwić wydajne korzystanie z witryny działającej w środowisku mobilnym. Z uwagi na mały obszar fizyczny należy priorytetowo potraktować funkcje usługi lub witryny. Nie ma miejsca na dodatkowe i rozpraszające aspekty projektu lub drugorzędne funkcje, zwłaszcza jeśli utrudniają one użytkownikom szybkie ukończenie podstawowego zadania (które czasem trzeba wykonać w pośpiechu lub w niedogodnym miejscu). Myślenie przede wszystkim o tych użytkownikach pomaga projektantom i programistom skoncentrować się na tym, co sprawia, że aplikacja jest wartościowa.

Podejście „zacznij od wersji mobilnej" może wydawać się nieodpowiednie, jeśli podstawowym celem jest przeniesienie do internetu mobilnego materiałów przechowywanych w systemie CMS. W końcu system już udostępnia świetną witrynę stacjonarną. Jednak każda witryna przechodzi przez etap odświeżania i zmian w projekcie. Kiedy już nauczysz się tworzyć witryny mobilne (a użytkownicy

mobilni zaczną stanowić duży odsetek wszystkich odwiedzających), bez wątpienia nadejdzie moment, w którym zechcesz postawić ich potrzeby i oczekiwania na równi z potrzebami użytkowników starszej wersji (lub nawet wyżej). Na bardzo praktycznym poziomie uwzględnianie najpierw użytkowników mobilnych jest bardzo przydatne przy projektowaniu interfejsów, które są *stopniowo wzbogacane* i reagują na cechy urządzenia po stronie klienta, co opisano w następnym podrozdziale.

Projektowanie interfejsów mobilnych

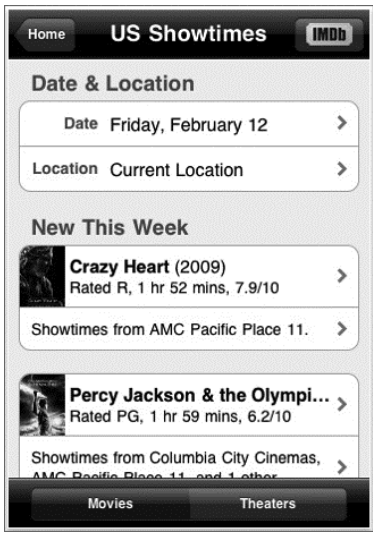
Wraz z pojawieniem się urządzeń przenośnych z ekranami dotykowymi nagle wystąpiło też uzasadnione zainteresowanie interfejsami użytkownika wykraczającymi poza interakcje oparte na myszy i klawiaturze. Autorzy witryny projektowanej i rozwijanej przed wprowadzeniem iPhone'a zakładali — co zupełnie zrozumiałe — że większość użytkowników korzysta z urządzeń wejścia dla komputerów stacjonarnych lub laptopów (klawiatury, myszy albo gładzika), a także dużych wyświetlaczy.

Oczywiście założenia te wpływały na to, w jaki sposób strony, witryny i aplikacje były rozwijane przez około 15 pierwszych lat internetu. Sieć WWW stała się oparta na dokumentach. Strony obejmowały stosunkowo dużą ilość odnośników, a przeglądarki — dodatkowe paski, co pozwalało udostępnić nawigację na całej stronie, odświeżanie itd. Sieć WWW i interfejsy przeglądarek, w których wyświetlano witryny, w niezwykle wysokim stopniu dostosowano do urządzeń wejścia i wyjścia dostępnych dla prawie każdego użytkownika. Odnośniki dostępne w dużej liczbie w zwijanych menu w pasku bocznym są bardzo łatwe do wskazania, przeczytania i kliknięcia, kiedy można posługiwać się precyzyjną myszą.

Jednak w urządzeniach mobilnych każde z tych założeń jest prawdopodobnie błędne. Niedostępna jest pełnowymiarowa klawiatura, którą użytkownik mógłby wykorzystać do wypełniania skomplikowanych formularzy. Ponadto z uwagi na mały ekran okno podglądu jest znacznie mniejsze niż wymiary większości witryn stacjonarnych. Użytkownik może korzystać z klawiszy sterujących fałszywym kursorem „myszy" w przeglądarce, jednak nie działają one tak precyzyjnie ani szybko jak rzeczywista mysz. Bardziej prawdopodobne jest to, że korzysta z urządzenia z ekranem dotykowym, co sprawia, iż w interfejsie użytkownika w ogóle nie istnieje kursor myszy. Przy poruszaniu szerokim palcem po ekranie trudne jest uzyskanie dokładności na poziomie pikseli, a interfejsy oparte na przytrzymywaniu kursora myszy nad elementami nie mają zastosowania.

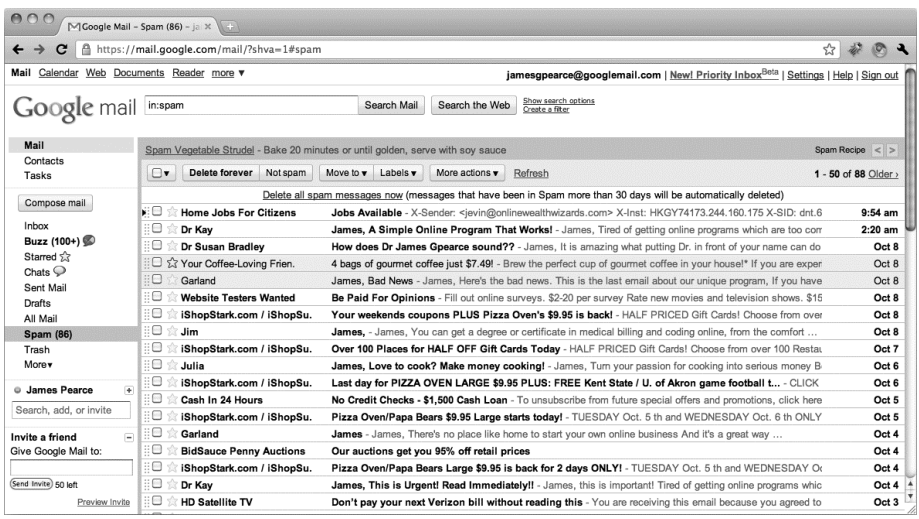
Społeczność internetowa i programiści stosunkowo powoli zaczęli zwracać uwagę na te drobne różnice oraz wykorzystywać nowe możliwości, jakie dawały urządzenia przenośne. Jednak bardzo duże zainteresowanie natywnymi aplikacjami wynikające z popularności sklepów z aplikacjami na urządzenia Apple'a i dla systemu Android sprawiło, że programiści zaczęli od nowa myśleć o interfejsach użytkownika i interakcji. Pojawiły się nowe i dostosowane wzorce projektowe dla aplikacji natywnych, takie jak specyficzna dla iPhone'a nawigacja przez przewijanie hierarchii, pionowe strony przypominające listy i proste paski narzędzi z przyciskami w dolnej części aplikacji. Na rysunku 9.10 pokazano natywną aplikację Internet Movie Database na iPhone'a, w której zastosowano niektóre z tych technik.

Wszystkie zmiany w standardowych założeniach na temat interfejsów wynikły z tego, że ekrany dotykowe stały się głównym urządzeniem wejścia. Za pomocą palca wskazującego można wygodnie poruszać się w poziomie po hierarchii stron w aplikacji, a przy użyciu kciuka można szybko przesunąć stronę w pionie jednym ruchem. Główne paski narzędzi aplikacji częściej znajdują się w dolnej części ekranu (gdzie można dotrzeć do nich za pomocą kciuka) niż na górnym pasku — tam zwykle widoczny jest tytuł strony i rzadziej używany przycisk *Wstecz*. W środowiskach opartych na ekranie dotykowym wprowadzono też nowe sztuczki z obszaru interfejsu użytkownika (na przykład przeciąganie listy chronologicznej w dół w celu jej odświeżenia). Gesty te często nie mają sensu w świecie zdominowanym przez mysz i klawiaturę.



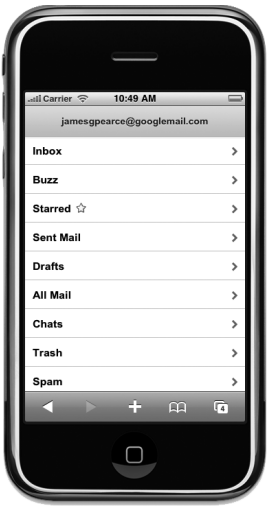
Rysunek 9.10. Natywna aplikacja w iPhone'ie

Dlatego choć pęd do rozwijania natywnych aplikacji klienckich na smartfony początkowo doprowadził do zmniejszenia zainteresowania technologiami opartymi na sieci WWW, pojawiło się w tym czasie wiele innowacji w interfejsach użytkownika. Prawie wszystkie innowacyjne podejścia dotyczą zarówno aplikacji i witryn rozwijanych za pomocą technologii internetowych, jak i rozwiązań tworzonych w językach natywnych. Zastanów się nad aplikacją Google Gmail (rysunek 9.11) i porównaj ją z wersją na iPhone'a (rysunki 9.12 i 9.13). Powinno być oczywiste, jak znacznie zmienił się projekt i interfejs użytkownika aplikacji mobilnych.

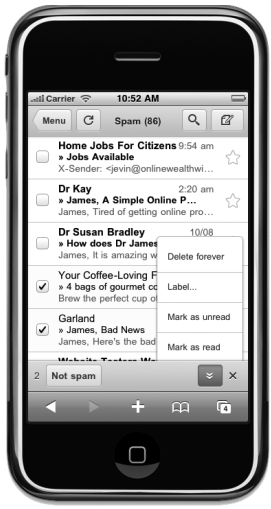


Rysunek 9.11. Stacjonarna witryna Google Gmail

Co ważniejsze, aplikacja w iPhone'ie jest wyraźnie oparta na projektach natywnych aplikacji. Podstawowa lista katalogów, która wcześniej znajdowała się w pasku bocznym w stacjonarnej wersji witryny, stała się punktem wejścia w formie listy. Przy wyświetlaniu zawartości konkretnego katalogu główny pasek menu do wykonywania operacji na wiadomościach jest umieszczony wzdłuż dolnej krawędzi ekranu, gdzie można operować kciukiem. Do wybierania (i oznaczania gwiazdkami)



Rysunek 9.12. Witryna Google Gmail w iPhone'ie



Rysunek 9.13. Witryna Google Gmail w iPhone'ie

wiadomości służą duże ikony dostosowane do wielkości palca. Wiele dodatkowych funkcji dostępnych w witrynie stacjonarnej usunięto. Przykładowo: za pomocą interfejsu mobilnego nie można zmienić ustawień dla e-maili lub skonfigurować nowych filtrów.

Jednak choć aplikacja mobilna została opracowana od nowa pod kątem użytkowników mobilnych, to przez zastosowanie tego samego schematu kolorów, spójnych ikon, etykiet i czcionek projektanci nie pozostawili wątpliwości, że nadal jest to serwis Gmail; witryna będzie natychmiast wyglądać znajomo dla użytkowników. Oczywiście nie ma recepty na niezawodne mobilne projekty i konwencje, a w obszarze tym cały czas pojawia się wiele innowacji. Sukces wzorców projektowych w dużym stopniu zależy też od urządzenia używanego do wyświetlania witryny oraz od wyświetlacza i urządzeń wejścia. Wystarczy powiedzieć, że inspirację można czerpać w równym stopniu z udanych natywnych aplikacji mobilnych, jak z innych witryn mobilnych.

Projekty mobilne oparte na kliencie

W społeczności projektantów witryn z dużym zainteresowaniem spotkała się jedna konkretna technika — **dostosowujące się projekty** (ang. *responsive design*). Nazwa pochodzi z artykułu Ethana Marcotte'a (*http://www.alistapart.com/articles/responsive-web-design/*), gdzie autor zasugerował, że serwer powinien zwracać jeden zestaw znaczników i rysunków dla strony, przy czym należy zastosować elastyczne i selektywne style po stronie klienta, aby dostosować układ do różnych wielkości ekranu i urządzeń.

Z jednej strony dla projektanta witryn, który dopiero wkracza w świat internetu mobilnego, podejście to jest bardzo kuszące. Nie trzeba pisać działającego po stronie serwera kodu do zwracania poszczególnym klientom różnych znaczników. Zmiany stylów są uzyskiwane wyłącznie na podstawie zapytań Media Query w stylach CSS. Przeciętny programista witryn zna tę technologię. Z drugiej strony zdaniem niektórych osób stosowanie dostosowujących się projektów to stosunkowo naiwny sposób na udostępnienie aplikacji w internecie mobilnym. Programista nie próbuje ograniczyć ilości kodu ani wielkości rysunków przesyłanych przez sieć do urządzenia przenośnego, a także z pewnością nie zastanawia się nad różnymi funkcjami, które mogą być potrzebne różnym grupom użytkowników.

Jeśli jednak jesteś zdecydowany na opracowanie jednego projektu witryny lub motywu, który działa stosunkowo dobrze w różnych przeglądarkach, warto znać to podejście. Można przynajmniej utworzyć różne projekty dla wersji mobilnej i stacjonarnej, a następnie zastosować dostosowujący się projekt w celu doprecyzowania wyglądu w przeglądarkach z tych dwóch ogólnych grup.

Wprowadzenie do zapytań Media Query

Kluczem do tworzenia dostosowujących się projektów są zapytania Media Query z języka CSS. Są one częścią specyfikacji CSS3 opracowanej przez W3C i umożliwiają projektantom określenie, w jakich warunkach stosowany ma być konkretny arkusz stylów (lub dany zestaw reguł z niego). Urządzenie klienckie (przy założeniu, że obsługuje zapytania Media Query, z czym radzi sobie większość systemów z zaawansowanych smartfonów) przetwarza zestaw prostych reguł, aby ustalić, które arkusze są odpowiednie.

Historycznie sposobem na wybiórcze stosowanie stylów CSS były typy mediów, które są dostępne od dawna i określają różne klasy urządzeń. Poniższy styl deklarowania odnośników w nagłówku dokumentu HTML pozwala odróżnić urządzenia typu `screen` od `handheld`.

```
<link rel="stylesheet" type="text/css" href="desktop.css" media="screen" />
<link rel="stylesheet" type="text/css" href="mobile.css" media="handheld" />
<link rel="stylesheet" type="text/css" href="print.css" media="print" />
```

Przeglądarki podające się za stacjonarne pobierają i stosują pierwszy arkusz stylów. Przeglądarki podające się za mobilne wykorzystują drugi arkusz. Niestety, starsze przeglądarki mobilne obsługują typy mediów na różnym poziomie, a współczesne urządzenia, na przykład iPhone, nie podają się w tym kontekście za przenośne. Trzeci arkusz jest stosowany w przeglądarkach stacjonarnych przy drukowaniu zawartości strony.

Wprowadzona niedawno specyfikacja zapytań Media Query umożliwia tworzenie precyzyjniejszych reguł pobierania i stosowania plików arkuszy stylów. Zapytania te są coraz bardziej niezawodnym i precyzyjnym sposobem wybiórczego określania stylów witryn. Oprócz ogólnego typu urządzenia można sprawdzić wiele właściwości sprzętu i wykorzystać je w kodzie warunkowym.

Do właściwości tych należą wielkość ekranu, orientacja i liczba obsługiwanych kolorów. Przykładowo: poniższy kod w nagłówku dokumentu oznacza, że urządzenie powinno wczytać różne arkusze stylów przeznaczone dla trzech przedziałów wielkości fizycznego ekranu.

```
<link rel="stylesheet" type="text/css" href="narrow.css"
  media="(max-device-width:320px)"
/>
<link rel="stylesheet" type="text/css" href="medium.css"
  media="(min-device-width:321px) and (max-device-width:480px)"
/>
<link rel="stylesheet" type="text/css" href="wide.css"
  media="(min-device-width:481px)"
/>
```

Oto właściwości, które można stosować w zapytaniach Media Query:

- **`width`**, **`height`** — wymiary okna podglądu przeglądarki (w pikselach);
- **`device-width`**, **`device-height`** — wymiary fizycznego ekranu (w pikselach);
- **`orientation`** — orientacja pionowa lub pozioma (określana na podstawie tego, czy wysokość jest większa od szerokości);
- **`aspect-ratio`**, **`device-aspect-ratio`** — stosunek wartości właściwości `width` do `height` i `device-width` do `device-height`;

- **color**, **color-index**, **monochrome** — liczba bitów na składową koloru dla ekranu urządzenia; wielkość tablicy kolorów (jeśli istnieje); liczba bitów na piksel w urządzeniach z ekranem monochromatycznym;
- **resolution** — gęstość pikseli w punktach na cal (dpi) lub punktach na centymetr (dpcm);
- **scan**, **grid** — proces skanowania stosowany w odbiorniku telewizyjnym; informacja, czy ekran obsługuje tylko wyświetlanie oparte na siatce, takie jak dla czcionek o stałej szerokości lub w terminalach tekstowych.

W kontekście internetu mobilnego prawdopodobnie najczęściej używane są właściwości `device-width`, `device-height` i `orientation`. Można dodać do wymiarów przedrostki `min-` i `max-`, aby określić w zapytaniach wartości graniczne. Warto wspomnieć także o specyficznej dla silnika WebKit właściwości z zapytań Media Query — `-webkit-min-device-pixel-ratio`. Właściwość tę można wykorzystać do zidentyfikowania wyświetlacza Retina o wyższej gęstości, stosowanego w iPhone'ie 4. Z uwagi na zgodność wstecz iPhone 4 skaluje jeden piksel ze stylów CSS na kratkę o wielkości 2 na 2 fizyczne piksele, a wartość wspomnianej właściwości dla tego urządzenia to 2.

W pokazanym przykładzie wykorzystano kilka arkuszy stylów do pokrycia w zasadzie ciągłego przedziału szerokości ekranu. Dlatego naturalne jest, że w ramach tych przedziałów należy zastosować *elastyczny* projekt, w którym elementy płynnie dopasowują się do ekranu (nie występuje tu sztywny układ oparty na siatce). Środkowy przedział obejmuje ekrany o szerokości od 321 do 480 pikseli. Przy sztywnym układzie strony dopasowanym do dolnego krańca tego przedziału na szerszych wyświetlaczach marnowana byłaby nawet jedna trzecia szerokości.

Oto prosty przykład. Poniższy plik jest prawie taki sam jak plik ze znacznikami wygenerowany przez kod do przełączania wersji witryny z rozdziału 7. (typ dokumentu to HTML5).


```
<!DOCTYPE html>
<html>
 <head>
  <title>Witaj</title>
 </head>
 <body>
  <h1>Witaj</h1>
  <div id="menu">
   <a href="index.php">Strona główna</a>
   <a href="page1.php">Strona 1.</a>
   <a href="page2.php">Strona 2.</a>
  </div>
  <div id="content">
   <p>
    <img src="lisa.jpg" alt="The Mona Lisa"/>
   </p>
   <p>
    Lorem ipsum dolor sit amet...
   </p>
   <p>
    Praesent sagittis...
   </p>
   <p>
    Curabitur rhoncus ipsum et...
   </p>
   <p>
    Donec sodales tristique auctor...
   </p>
```

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

Plik responsive/index.html

Przy użyciu elementów `<link>` w nagłówku można dodać arkusze stylów odpowiadające trzem przedziałom szerokości ekranu.

```
<link rel="stylesheet" type="text/css" href="narrow.css"
  media="(max-width:320px)"
/>
<link rel="stylesheet" type="text/css" href="medium.css"
  media="(min-width:321px) and (max-width:480px)"
/>
<link rel="stylesheet" type="text/css" href="wide.css"
  media="(min-width:481px)"
/>
```

Warto zauważyć, że zastosowano tu właściwość `width` zamiast `device-width`. Ta pierwsza dotyczy wielkości okna przeglądarki, a druga — fizycznej wielkości ekranu. Przy testowaniu zapytań Media Query na przeglądarce stacjonarnej przydatna jest właściwość `width`, pozwalająca zobaczyć zmiany stylów przy ręcznej zmianie wielkości okna. Stosowanie jej dla urządzeń przenośnych może początkowo wydawać się ryzykowne — przeglądarka z iPhone'a i inne przeglądarki oparte na oknie podglądu domyślnie przyjmują, że okno podglądu ma szerokość 980 pikseli (co pozwala na poprawną obsługę starszych witryn stacjonarnych), dlatego można podejrzewać, że style dla mniejszych szerokości nigdy nie zostaną aktywowane.

Na szczęście można zmienić wielkość okna podglądu używanego przez przeglądarkę dla danej witryny. Służy do tego właściwość `viewport` w sekcji `<head>` kodu znaczników.

```
<meta name="viewport" content="width=600px" />
```

Można też zastosować poniższą deklarację do ustawienia na stałe okna podglądu na wielkość równą ekranowi fizycznemu.

```
<meta name="viewport" content="
  width=device-width,
  initial-scale=1.0
" />
```

Teraz, przynajmniej w orientacji pionowej, właściwości `width` i `device-width` mają tę samą wartość. Jednak stosowanie właściwości `width` zamiast `device-width` jest poważną zaletą w smartfonach wykrywających orientację ekranu. Po zmianie orientacji właściwość `device-width` przyjmuje nową wartość (480 zamiast 320 pikseli), natomiast fizyczna szerokość (przynajmniej w iPhone'ie, w którym domyślna jest orientacja pionowa) nadal wynosi 320 pikseli. Ponieważ przeważnie pożądane jest, aby dostosowujący się projekt pasował do szerszego ekranu w orientacji poziomej, można użyć właściwości `width` w zapytaniu Media Query, a ponadto wykorzystać deklarację `viewport`, aby zachować obie omawiane właściwości przeglądarki pod kontrolą. Jeśli chcesz utworzyć *naprawdę* bezpieczne rozwiązanie, możesz posunąć się o krok dalej i wyłączyć możliwość przybliżania i oddalania obrazu przez użytkownika, a także zagwarantować, że przełączanie się między trybem poziomym a pionowym nie prowadzi do zmiany skalowania okna podglądu.

```
<meta name="viewport" content="
  width=device-width,
  initial-scale=1.0,
  minimum-scale=1.0,
```

```
    maximum-scale=1.0,
    user-scalable=false
" />
```

Dostosowujący się projekt

Wróćmy do arkuszy stylów. Sposób podziału przedziałów wielkości ekranów urządzeń może zależeć od celów projektowych, jednak trzy stosowane tu przedziały pokrywają się mniej więcej z układem pionowym w urządzeniach przenośnych, układem poziomym w urządzeniach przenośnych i ekranami urządzeń stacjonarnych. Teoretycznie plik *narrow.css* obsługuje dowolny ekran o szerokości 320 pikseli lub mniejszej. W praktyce prawdopodobnie będzie przeznaczony tylko dla smartfonów o szerokości 320 pikseli, ponieważ większość zwykłych i uboższych w funkcje telefonów (które zazwyczaj mają węższe ekrany) w ogóle nie obsługuje zapytań Media Query i arkuszy stylów.

Aby witryna wyglądała poprawnie w urządzeniach, które nie obsługują ustawionych warunków, trzeba udostępnić rezerwowy arkusz stylów stosowany w każdej sytuacji. Można go podać na pierwszym miejscu listy i umieścić w nim proste style przeznaczone dla docelowych urządzeń przenośnych o mniejszych możliwościach. Jest to także dobre miejsce na zapisanie stylów wspólnych dla wszystkich urządzeń (na przykład schematu kolorów obowiązującego w witrynie).

Następne zagadnienie jest bardzo ważne. Jeśli planujesz stosować zapytania Media Query do uproszczenia obecnej witryny stacjonarnej pod kątem urządzeń przenośnych, spowodujesz problemy we wszystkich urządzeniach, które nie rozumieją takich zapytań. Poniższy kod znaczników jest bardzo złym rozwiązaniem.

```
<link rel="stylesheet" type="text/css" href="huge.css" media="all"/>
<link rel="stylesheet" type="text/css" href="lite.css"
  media="(max-device-width:480px)"
/>
```

Duży (prawdopodobnie stacjonarny) arkusz stylów jest tu wczytywany we wszystkich urządzeniach, a tylko te, które przetwarzają zapytania Media Query (i mają odpowiedni ekran), zastosują drugi arkusz. Jednak w starszych i mniej zaawansowanych urządzeniach dostępny będzie tylko styl *huge.css*, którego bez wątpienia nie zaprojektowano na potrzeby urządzeń z niższej półki.

Oto kolejny przykład myślenia w kategoriach „najpierw wersja mobilna”. Idealne rozwiązanie to utworzenie prostego, ale opartego na stylach i funkcjonalnego interfejsu na potrzeby najmniej zaawansowanych urządzeń oraz zastosowanie zapytań Media Query w celu utworzenia bogatszych, bardziej rozbudowanych interakcji pod kątem urządzeń o szerszych ekranach i potrafiących interpretować wspomniane zapytania. Ta technika to **stopniowe wzbogacanie** (ang. *progressive enhancement*). Programiści witryn stacjonarnych często ją stosują, kiedy muszą poradzić sobie z różnorodnymi możliwościami przeglądarek (choć trzeba przyznać, że nazwa ta często pojawia się w kontekście języka JavaScript, a nie stylów CSS).

Przed stylami specyficznymi dla szerokości ekranu trzeba więc wstawić dodatkowy odnośnik do bezpiecznego, podstawowego arkusza stylów.

```
<link rel="stylesheet" type="text/css" href="basic.css" media="all"/>
<link rel="stylesheet" type="text/css" href="narrow.css"
  media="(max-width:320px)"
/>
<link rel="stylesheet" type="text/css" href="medium.css"
  media="(min-width:321px) and (max-width:480px)"
/>
```

```
<link rel="stylesheet" type="text/css" href="wide.css"
  media="(min-width:481px)"
/>
```

W pliku *basic.css* można umieścić informacje o kolorze, czcionkach i innych podstawowych spójnych aspektach projektu. Na podstawie tego stylu można oczywiście tworzyć dodatkowe arkusze, jednak podstawowe style powinny być niezależne ze względu na prostsze urządzenia, które mogą nie otrzymać dodatkowych stylów.

Do pobrania w witrynie helion.pl

```
body {
  background:#ccc;
  font-family:sans-serif;
}

h1 {
  margin:0;
}

#menu {
  padding:10px;
  background:#000;
  color:#fff;
  font-weight:bold;
  font-size:smaller;
}

#menu a {
  color:#fff;
}

#content img {
  width:100%;
}
```

Plik responsive/basic.css

W kodzie nie ma nic szczególnie skomplikowanego, ale to jeszcze nie wszystko. Jest to prosty zestaw stylów, który powinien działać w prawie każdym urządzeniu przenośnym. W starszych urządzeniach z systemem Windows Mobile zastosowane zostaną tylko te style (ponieważ system nie obsługuje zapytań Media Query). Na rysunku 9.14 przedstawiono akceptowalne efekty działania tego rozwiązania.

Przed opracowaniem dalszych stylów warto zobaczyć, jak strona wygląda w przeglądarce w iPhone'ie w orientacji pionowej, co pokazano na rysunku 9.15.

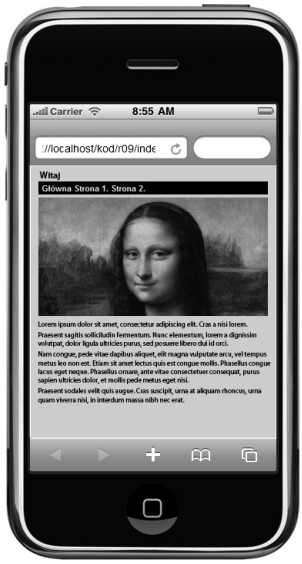
Najoczywistszą wadą jest mała wielkość tekstu (choć rysunek jest rozciągnięty na szerokość ekranu). Wynika to z okna podglądu w iPhone'ie. Widoczna jest strona rozciągnięta do wirtualnego ekranu o szerokości 980 pikseli, a następnie pomniejszona, tak aby pasowała do fizycznego ekranu o szerokości 320 pikseli. Dla rysunku zastosowano styl `width:100%`, co „wirtualnie" oznacza prawie 980 pikseli na szerokość, dlatego obrazek wygląda na duży, jednak tekst jest tu nieczytelny.

Można rozwiązać ten problem przez ustawienie okna podglądu tak, aby od początku obowiązywało odwzorowanie jeden do jednego względem wielkości ekranu.

```
<meta name="viewport" content="width=device-width" />
```

Nie warto ryzykować podawania dla okna podglądu bezwzględnej szerokości w pikselach (320). Nie zapewnia to zgodności z przyszłymi, coraz większymi ekranami, ani nie pozwala radzić sobie z orientacją poziomą nawet na omawianym urządzeniu. Obecny wygląd tej samej strony pokazano na rysunku 9.16.

Rysunek 9.14. Strona z prostym stylem w urządzeniu z systemem Windows Mobile



Rysunek 9.15. Strona z prostym stylem w iPhone'ie

Rysunek 9.16. Ta sama strona po zmianie stylów

Teraz można dodać ciekawe style do pliku *narrow.css*. Przykładowo: można przyjąć, że urządzenie o niewielkim ekranie, które potrafi przetwarzać zapytania Media Query, ma wyświetlacz dotykowy. Dlatego odnośniki powinny być większe i łatwiejsze do obsługi za pomocą dotyku.

Do pobrania w witrynie helion.pl

```
h1 {
  text-align:center;
}

#menu {
  text-align:center;
  background:none;
}

#menu a {
  font-size:medium;
  text-decoration:none;
  padding:5px 10px;
  margin:0;
  border:2px solid #fff;
  background:#000;
}
```

Plik responsive/narrow.css

Tu tytuł jest umieszczony pośrodku okna podglądu o szerokości 320 pikseli, a styl tła usunięto z menu i powiązano z każdym odnośnikiem. Ponadto dodano ramkę, aby odnośniki bardziej przypominały przyciski. Efekt, widoczny na rysunku 9.17, jest spójny ze stylem witryny (trzeba przyznać, że dość zwyczajnym i monochromatycznym), jednak lepiej dopasowany do szerokości ekranu. Nie trzeba wspominać, że można poeksperymentować ze stylami dla silnika WebKit, aby utworzyć bardziej eleganckie przyciski.

Rysunek 9.17. Strona ze stylami dostosowanymi do ekranu dotykowego

Teraz możesz obrócić urządzenie, aby przystąpić do pracy nad następnym stylem, z pliku *medium.css*. Wcześniej jednak warto przyjrzeć się obecnemu wyglądowi strony (rysunek 9.18). Warto zauważyć, że szerokość okna podglądu wzrosła do 480 pikseli (a w zapytaniach Media Query używana jest właściwość `width`), dlatego utracono efekt przycisków z pliku *narrow.css*.

Rysunek 9.18. Strona w orientacji poziomej

Można przenieść menu na lewą stronę ekranu i lepiej wykorzystać szerokość wyświetlacza. 480 pikseli prawdopodobnie oznacza, że używane jest urządzenie z ekranem dotykowym, dlatego ponownie można zastosować style upodabniające odnośniki do przycisków.


```
h1 {
  border-bottom:1px solid #fff;
}

#menu {
  position:absolute;
  left:8px;
  width:146px;
  padding:0;
  background:none;
}

#menu a {
  display:block;
  text-align:center;
```

```
  font-size:medium;
  text-decoration:none;
  margin:8px 0;
  padding:5px 10px;
  border:2px solid #fff;
  background:#000;
}

#content {
  position:absolute;
  left:160px;
  padding-left:7px;
  border-left:1px solid #fff;
}

#content img {
  max-width:304px;
  display:block;
}

#content p:first-child {
  margin-top:8px;
}
```

Plik responsive/medium.css

Tu menu umieszczono po lewej stronie i ustawiono jego szerokość na 146 pikseli. Odnośniki w menu są elementami blokowymi, dlatego pojawiają się w kolumnie. Odpowiednio dostosowano marginesy, aby odnośniki były odpowiednio oddalone od siebie.

Obszar z treścią (główną część strony) przeniesiono na prawo i określono dla niego taką samą szerokość, jak w orientacji pionowej (przynajmniej na ekranach o szerokości 480 pikseli). Nie jest to konieczne, jednak zapewnia równowagę w projekcie. W rzeczywistości szerokość jest tu nieokreślona, dlatego układ pozostaje elastyczny. Jeśli użytkownik wyświetli stronę na ekranie o szerokości na przykład 400 pikseli, kolumna z treścią zostanie zwężona, a tekst odpowiednio się dopasuje. Efekt pokazano na rysunku 9.19.

Rysunek 9.19. Strona ze stylami dostosowanymi do orientacji poziomej

Jeśli zauważysz, że stosujesz te same style w wielu plikach (tak jak tutaj), powinieneś umieścić kod tych stylów w jednym miejscu, tak aby go nie powielać. Rozbicie stylów na kilka plików powoduje jednak, że urządzenie musi zgłaszać dodatkowe żądania HTTP.

Jest to dobre miejsce do tego, aby pokazać inny sposób stosowania zapytań Media Query. Wyrażenia można umieszczać *wewnątrz* arkusza stylów za pomocą prostej struktury zagnieżdżonej objętej zapytaniem.

```
@media (min-width:321px) {
  body {
    background:red;
  }
}
```

Za pomocą tej techniki można połączyć arkusze stylów dla wąskich i średnich wyświetlaczy. Wspólne style (na przykład przycisków) przeniesiono na początek arkusza, a układ specyficzny dla ekranów o różnych przedziałach umieszczono w dwóch blokach selektorów pogrupowanych według zapytań Media Query.


```
#menu {
  background:none;
}

#menu a {
  font-size:medium;
  text-decoration:none;
  padding:5px 10px;
  margin:0;
  border:2px solid #fff;
  background:#000;
}

@media (max-width:320px) {
  h1 {
    text-align:center;
  }

  #menu {
    text-align:center;
  }

  #menu a {
    margin:0;
  }
}

@media (min-width:321px) {
  h1 {
    border-bottom:1px solid #fff;
  }

  #menu {
    position:absolute;
    left:8px;
    width:146px;
    padding:0;
  }

  #menu a {
    display:block;
    text-align:center;
    margin:8px 0;
  }

  #content {
    position:absolute;
```

```
    left:160px;
    padding-left:7px;
    border-left:1px solid #fff;
  }

  #content img {
    max-width:304px;
  }

  #content p:first-child {
    margin-top:8px;
  }
}
```

Plik responsive/narrow-medium-wide.css

Plik można dołączyć do strony przez zastąpienie wcześniejszych deklaracji `<link>` poniższymi.

```
<link rel="stylesheet" type="text/css" href="basic.css" media="all"/>
<link rel="stylesheet" type="text/css" href="narrow-medium-wide.css"
  media="(min-width:0px)"
/>
```

Dostępne są podstawowy arkusz stylów i połączony arkusz stylów dla dostosowującego się projektu, z warunkiem opartym na zapytaniach Media Query. Choć warunek jest spełniony dla wszystkich urządzeń obsługujących wspomniane zapytania, takie podejście to dobry sposób na wykluczenie pozostałych urządzeń i upewnienie się, że nie próbują zastosować reguł z drugiego arkusza stylów.

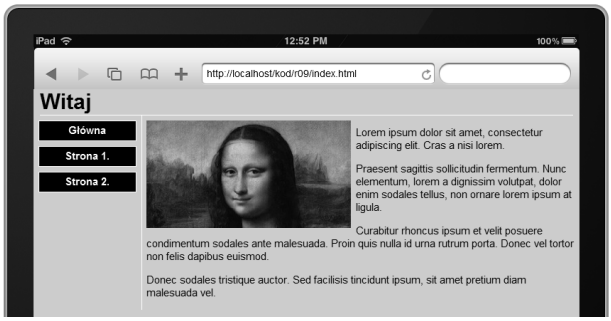
Teraz można łatwo dodać do pliku z połączonymi stylami ostatni blok, przeznaczony dla przedziału obejmującego duże ekrany (prawdopodobnie z tabletów lub komputerów stacjonarnych), o szerokości 481 pikseli lub większej. Jedyne różnice to wyświetlanie rysunku wewnątrzwierszowo, tak aby tekst go opływał, i ograniczenie szerokości elastycznego układu, żeby wiersze tekstu nie były zbyt długie na stacjonarnych monitorach panoramicznych lub na tabletach w orientacji poziomej.

```
@media (min-width:481px) {
  #content {
    max-width:640px;
  }

  #content img {
    padding:0 8px 8px 0;
    display:inline;
    float:left;
  }
}
```

Wygląd stylu dla większych wyświetlaczy pokazano na rysunku 9.20. Widać tu iPada w orientacji pionowej (szerokość wynosi 768 pikseli).

Skalowanie rysunków

Ostatnie zagadnienie z obszaru dostosowujących się projektów opartych na stylach CSS dotyczy kwestii, na którą może już zwróciłeś uwagę. Nie uwzględniono tu wielkości rysunków przesyłanych do urządzeń. Obraz Mona Lizy jest statycznym zasobem, plikiem *lisa.jpg* o wymiarach 304 na 159 pikseli. Zajmuje on skromne 20 kilobajtów, co nie jest problemem przy przesyłaniu danych przez sieć mobilną niezależnie od tego, czy urządzenie przenośne dodatkowo zmniejsza obrazek.



Rysunek 9.20. Strona ze stylem dla urządzeń o większych ekranach

Łatwo jednak wyobrazić sobie sytuację, w której pożądane jest skalowanie większych, bardziej szczegółowych rysunków w dużo większym przedziale wymiarów. Aby rysunek był szczegółowy przy wysokich rozdzielczościach, pierwotny plik musi mieć wymiary przynajmniej równe wymiarom wyświetlacza, co oczywiście oznacza zwiększenie wielkości pliku, który urządzenia o mniejszych ekranach muszą niepotrzebnie pobierać (nie wspominam tu nawet o mocy obliczeniowej potrzebnej do zmiany wielkości rysunku „w locie").

Pierwszy pomysł może polegać na przygotowaniu zestawu różnych rysunków, które można wybiórczo pobierać na urządzenia. Jednak w tym podrozdziale pokazano, jak dostosować się do różnych urządzeń po stronie klienta, bez polegania na mechanizmie wykrywania przeglądarek po stronie serwera. Ponieważ używany jest statyczny kod znaczników, nie jest oczywiste, jak wyświetlać rysunki o różnej wielkości i jednocześnie wydajnie korzystać z sieci.

Bardzo atrakcyjną możliwością jest zastosowanie zewnętrznego serwisu do zmieniania wielkości rysunków. tinySrc, internetowy serwis tego rodzaju, można wykorzystać do skalowania obrazków na potrzeby dostosowującego się projektu dla urządzeń przenośnych. Aby użyć interfejsu API serwisu tinySrc, wystarczy dodać przedrostek tinySrc do adresu URL całego rysunku. Wyobraź sobie, że w kodzie znaczników znajduje się następujący znacznik.

```
<img
  src="http://mysite.com/lisa.jpg"
  alt="The Mona Lisa
"/>
```

Wystarczy zastąpić ten znacznik poniższym kodem.

```
<img
  src="http://i.tinysrc.mobi/http://mysite.com/lisa.jpg"
  alt="The Mona Lisa
"/>
```

Serwis tinySrc korzysta z bazy danych Device Atlas do zmiany wielkości rysunków w taki sposób, aby pasowały do dowolnego urządzenia zgłaszającego żądanie. Możliwe, że zauważyłeś, iż przy wyświetlaniu obrazka na iPhone'ie wokół strony występuje margines 8 pikseli, dlatego grafika nie powinna mieć szerokości równej dokładnie 320 pikselom (jest to fizyczna szerokość wyświetlacza iPhone'a). Oparty na adresach URL interfejs API serwisu tinySrc umożliwia też określenie, że od nowej wielkości rysunku należy odjąć marginesy.

```
<img
  src="http://i.tinysrc.mobi/-16/http://mysite.com/lisa.jpg"
  alt="The Mona Lisa
"/>
```

Kompletna dokumentacja serwisu tinySrc znajduje się pod adresem *http://tinysrc.net*.

Projekty mobilne oparte na serwerze

W poprzednim podrozdziale dowiedziałeś się, że ten sam kod znaczników z jednym podstawowym stylem i dodatkowymi stylami opartymi na zapytaniach Media Query pozwala opracować projekt na tyle elastyczny, że będzie działał prawidłowo dla ekranów o różnej szerokości. Warto jednak powtórzyć stwierdzenie z innego miejsca książki — tworząc projekt mobilny, prawdopodobnie należy dążyć do czegoś więcej.

Uwzględnianie różnorodności

Możliwe, że punkt wyjścia (istniejąca witryna stacjonarna lub obecny motyw) jest zbudowany w taki sposób, że nie da się zastosować dostosowującego się projektu. Idealnym rozwiązaniem jest projekt o elastycznej szerokości, jednak od kilku lat znacznie popularniejsze są projekty o stałej szerokości. Trudno wyobrazić sobie przekształcenie szablonu o ścisłych ograniczeniach na wersję odpowiednią dla węższego, mobilnego interfejsu.

Jednak znacznie ważniejsze od tych praktycznych kwestii jest to, że programista poświęcił bardzo mało uwagi bardziej podstawowym zmianom, na które zasługują użytkownicy mobilni. Przesyłając ten sam kod znaczników do zaawansowanego telefonu (jedna skrajność) i 27-calowego stacjonarnego monitora panoramicznego (druga skrajność), programista zakłada, że użytkownicy każdego z tych urządzeń oczekują dokładnie identycznych witryn, usług, architektur informacji itd. W praktyce całościowa strategia rozwijania rozwiązań mobilnych oparta jest na dostarczaniu tego, czego oczekują różne grupy użytkowników. Przecież jedna osoba może w pośpiechu iść chodnikiem, natomiast druga — wygodnie siedzieć przed zamontowanym na ścianie monitorem. Oczywiste jest, że wymagania obu tych osób są inne! W wielu sytuacjach różnice w funkcjach i usługach, które trzeba udostępniać w skrajnie odmiennych urządzeniach i kontekstach, są zbyt duże, aby można je było uwzględnić za pomocą samych stylów CSS i zapytań Media Query. Trzeba wbudować „inteligencję" także po stronie serwera, tak aby mógł wysyłać do poszczególnych urządzeń różne materiały.

Choć w tym krótkim rozdziale nie opisano ponownie przełączania między wersjami stacjonarną i mobilną (temat ten omówiono dość dokładnie w rozdziale 7.), pokazano, co można zrobić w kontekście projektu, jeśli programista *zdecydował się* wykrywać urządzenia po stronie serwera. Skoncentrowano się przy tym na dostosowaniu projektu do różnych grup urządzeń przenośnych.

Projektowanie pod kątem grup urządzeń

We wcześniejszych rozdziałach poruszono technikę podziału docelowych urządzeń na grupy. Przede wszystkim, co oczywiste, należy ustalić, czy urządzenie żądające strony z witryny jest mobilne. Przez zastosowanie wykrywania po stronie serwera, zwłaszcza w połączeniu z bazą danych urządzeń, można stosunkowo łatwo opracować różne interfejsy i wykrywać, jakiego rodzaju urządzenie uzyskuje dostęp do witryny.

Przykładowo: można ocenić różne rodzaje przeglądarek i przygotować dla każdej grupy inną wersję motywu mobilnego. W tabeli 9.1 przedstawiono możliwy podział na grupy.

Urządzenia z tabeli można łatwo podzielić na mniejsze grupy, zwłaszcza jeśli dokładnie wiadomo, które funkcje urządzeń witryna powinna obsługiwać. Przykładowo: jeżeli programista chce w dużym zakresie stosować w witrynie AJAX-a, obsługa odpowiednich technologii będzie równie ważnym kryterium podziału na grupy, co przetwarzanie różnych rodzajów znaczników.

Tabela 9.1. Proste pogrupowanie urządzeń przenośnych

Grupa	Obsługiwane technologie	Przykłady
A	HTML5, CSS3, JavaScript, prawdopodobnie duży obracany ekran dotykowy	Przeglądarki z systemów iOS, Android, webOS, Symbian S60 wersja 5
B	XHTML, CSS2, JavaScript (w ograniczonym zakresie), prawdopodobnie ekran średniej wielkości	BlackBerry wersja 5 i starsze, Windows Mobile wersja 6.5 i starsze
C	XHTML-MP, CSS (w ograniczonym zakresie), brak obsługi JavaScriptu, prawdopodobnie mały ekran	Większość urządzeń wyprodukowanych między 2004 a 2007 rokiem
D	Brak obsługi HTML-a lub XHTML-a, prawdopodobnie mały ekran	Starsze urządzenia z obsługą protokołu WAP sprzed 2004 roku

Na tym etapie warto powiedzieć, że bardzo trudno jest zbudować witrynę działającą w spójny sposób na każdym możliwym urządzeniu i we wszystkich przeglądarkach. Zawsze będą istnieć starsze urządzenia (na przykład grupa D z wcześniejszej tabeli) o ograniczonych możliwościach i coraz mniejszej liczbie użytkowników. Nie warto poświęcać czasu na budowanie lub modyfikowanie witryny pod kątem takich urządzeń. Należy podjąć świadomą decyzję i pominąć projektowanie pod kątem ostatniej grupy oraz zrezygnować z obsługi należących do niej urządzeń.

Niezalecane jest też popadanie w drugą skrajność i obsługiwanie tylko jednej grupy urządzeń. Projekt witryny działający na urządzeniach z tylko jednej konkretnej grupy może być ciekawy i łatwy do opracowania, jednak jeśli nie funkcjonuje poprawnie we wszystkich innych urządzeniach, pozbawiasz się milionów potencjalnych użytkowników, którzy nie korzystają z danego modelu. Może się wydawać, że wszyscy Twoi znajomi ze społeczności programistów witryn używają na przykład iPhone'a firmy Apple (a Ty sam możesz korzystać z jego emulatora na potrzeby tej książki!), jednak z pewnością nie oznacza to, iż wszystkie osoby odwiedzające Twoją witrynę też mają to urządzenie. Użytkownicy nie podziękują Ci za lenistwo i nieuwzględnianie ich potrzeb, co uniemożliwiło dostęp do oferowanych usług. Nieważne, ilu użytkowników mobilnych korzysta z danego modelu telefonu — zawsze będą miliardy innych, którzy używają odmiennych urządzeń.

Wykrywanie, do której grupy należy urządzenie, nie jest nauką ścisłą. Proces ten zależy w dużym stopniu od wybranych kryteriów podziału na grupy. Jednak każda wartościowa baza danych urządzeń (na przykład WURFL lub DeviceAtlas) obejmuje właściwości określające obsługę języków znaczników, a także inne dane, które można wykorzystać do podziału na grupy. Wszystkie te informacje są otrzymywane na podstawie nagłówka `user-agent` przeglądarki z urządzenia. Przykładowo: jeśli zainstalowałeś bazę DeviceAtlas, poniższy kod pomoże Ci podzielić zgłaszające żądania urządzenia przenośne na cztery grupy, od A do D. E oznacza nieznany sprzęt, a null — urządzenie, które nie jest przenośne.

Do pobrania w witrynie helion.pl

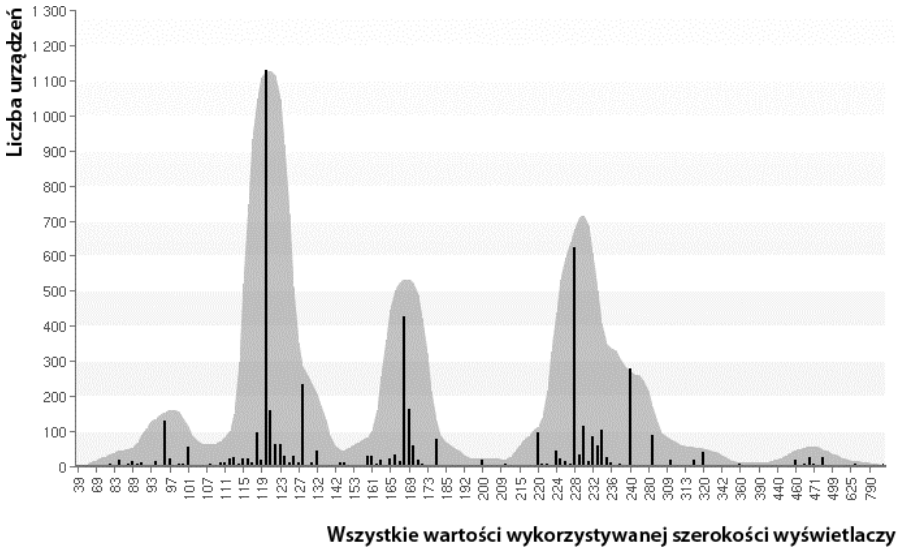
```
<?php

require_once '../lib/da/Api.php';
$tree = Mobi_Mtld_DA_Api::getTreeFromFile(
  "../lib/da_resources/deviceatlas.json"
);
$user_agent = get_http_header('User-agent');

print "Przeglądarka należy do grupy: " . device_group($tree, $user_agent);

function device_group($tree, $user_agent) {
  if (!device_property($tree, $user_agent, 'mobileDevice', 0)) {
    return null;
```

```
  }
  if (device_property($tree, $user_agent, 'touchScreen', 0)) {
    return 'A';
  }
  if (device_property($tree, $user_agent, 'markup.xhtmlBasic10', 0)) {
    return 'B';
  }
  if (device_property($tree, $user_agent, 'markup.xhtmlMp10', 0)) {
    return 'C';
  }
  if (device_property($tree, $user_agent, 'markup.wml1', 0)) {
    return 'D';
  }
  return 'E';
}

function device_property($tree, $user_agent, $property, $default) {
  try {
    $value = Mobi_Mtld_DA_Api::getProperty($tree, $user_agent, $property);
    if (is_null($value)) {
      $value = $default;
    }
  } catch (Exception $e) {
    $value = $default;
  }
  return $value;
}

function get_http_header($name, $original_device=true, $default='') {
  if ($original_device) {
    $original = get_http_header("X-Device-$name", false);
    if ($original!=='') {
      return $original;
    }
  }
  $key = 'HTTP_' . strtoupper(str_replace('-', '_', $name));
  if (isset($_SERVER[$key])) {
    return $_SERVER[$key];
  }
  return $default;
}

?>
```

Plik da.php

Działanie kodu jest bardzo proste — program wykrywa grupy na podstawie obsługi języków znaczników. Po ustaleniu, do której z głównych grup należy przeglądarka, można wskazać warstwie obsługującej motyw lub style witryny odpowiednie pliki szablonu. Warto zauważyć, że użyto funkcji pomocniczej `device_property` do obsługi wartości domyślnych w sytuacjach, kiedy w bazie danych urządzeń wartość konkretnej właściwości jest nieznana. Ponownie wykorzystano też roboczą funkcję do sprawdzania nagłówków.

Oprócz grupowania urządzeń na podstawie konkretnych wartości właściwości można też utworzyć kategorie oparte na wartościach ciągłych właściwości, takich jak wielkość ekranu. W rozdziale 2. napisano, że szerokość ekranu urządzenia może wskazywać na rodzaj i ogólne możliwości urządzenia. Na rysunku 9.21 pokazano rozkład liczby urządzeń o określonych szerokościach ekranu.



Rysunek 9.21. Liczba urządzeń o określonych szerokościach ekranu

Jeśli zamierzasz oprzeć się na tym rozkładzie, uzasadnione będzie utworzenie projektów dla ogólnych grup pokazanych w tabeli 9.2.

Tabela 9.2. Grupy urządzeń przenośnych oparte na szerokości ekranu

Grupa	Kryterium	Przykłady
A+	Ponad 400 pikseli	Smartfony (w orientacji poziomej), komunikatory, tablety; prawdopodobnie zaawansowane i z ekranami dotykowymi
A	Od 240 do 400 pikseli	Smartfony (w orientacji pionowej); prawdopodobnie zaawansowane i z ekranami dotykowymi
B+	Od 200 do 240 pikseli	Smartfony; ze średniej i wyższej półki, bez ekranów dotykowych
B	Od 140 do 200 pikseli	Urządzenia ze średniej półki i nowe z niższej
C	Od 110 do 140 pikseli	Starsze urządzenia ze średniej i niższej półki
D	Poniżej 110 pikseli	Starsze urządzenia z obsługą protokołu WAP

Jest mało prawdopodobne, że podział ten idealnie pasuje do wspomnianych wcześniej grup opartych na językach HTML, CSS i JavaScript, może być jednak sensownym przybliżeniem. Przykładowo: poza kilkoma wyjątkami urządzenia o dużej liczbie pikseli na szerokość są nowe i z wyższej półki (a tym samym mają ekran dotykowy). Zaletą stosowania prostej właściwości w rodzaju szerokości ekranu jest szybkość i łatwość oceny urządzenia na podstawie bazy danych w jednej operacji wyszukiwania (uzyskaną wartość można później wykorzystać przy generowaniu stron). Inna korzyść jest taka, że jeśli projektujesz elementy graficzne (na przykład logo i ekrany powitalne) dla różnych grup urządzeń, wiesz, jakie są podstawowe ograniczenia wielkości ekranu w każdej grupie.

Z podziałem na grupy zetkniesz się ponownie w kontekście konkretnych systemów CMS w dalszych rozdziałach. Autorzy kilku wtyczek i modułów dostępnych w systemach WordPress oraz Drupal uwzględnili potrzebę stosowania więcej niż dwóch motywów dla interfejsów mobilnych i umożliwili podział zbioru urządzeń na wiele grup.

Łączenie podejść

Kiedy zastanowisz się nad grupowaniem urządzeń na podstawie ekranu, do głowy może przyjść Ci ciekawy pomysł. Wcześniej w rozdziale opisano stosowanie dostosowującego się projektu jako sposobu na zapewnienie eleganckiego układu witryny na ekranach o różnej wielkości. Jeśli umieścisz urządzenia w odrębnych grupach opartych na ciągłej zmiennej (tak jak w ostatnim podziale), musisz zapewnić obsługę urządzeń z obu skrajnych punktów każdego przedziału. Czy możliwe jest połączenie obu podejść?

Oczywiście odpowiedź brzmi: tak. W inteligentnym podejściu do projektowania pod kątem wielu urządzeń należy harmonijnie stosować techniki działające po stronie serwera i klienta. Choć może to wymagać koordynacji między dwoma wspomnianymi obszarami witryny i procesu jej rozwijania, podejście jest bardzo sprytne i daje duże możliwości. Pomyśl o tym jak o sposobie na przeniesienie wybranych zapytań Media Query na serwer. Przyjrzyj się poniższemu kodowi, który wykonuje właśnie to zadanie.

Do pobrania w witrynie helion.pl

```
<?php

require_once '../../lib/da/Api.php';
$tree = Mobi_Mtld_DA_Api::getTreeFromFile(
  "../../lib/da_resources/deviceatlas.json"
);

$user_agent = get_http_header('User-agent');

$device_group = device_group($tree, $user_agent);
$touch_screen = device_property($tree, $user_agent, 'touchScreen', 0);

function device_group($tree, $user_agent) {
  if (device_property($tree, $user_agent, 'mobileDevice', 0)) {
    $default_width=160;
  } else {
    $default_width=640;
  }
  $width = device_property(
    $tree, $user_agent, 'displayWidth', $default_width
  );
  if ($width < 241) {
    return 'uproszczona';
  } elseif ($width < 321) {
    return 'wąska';
  } elseif ($width < 481) {
    return 'średnia';
  } else {
    return 'szeroka';
  }
}

...

?><!DOCTYPE html>
<html>
 <head>
  <title>Witaj</title>
    <link rel="stylesheet" type="text/css" href="basic.css" media="all"/>
    <link rel="stylesheet" type="text/css" href="<?php
```

```
        print $device_group;
      ?>.css" media="all"/>
      <?php if ($touch_screen) { ?>
        <meta name="viewport" content="
          width=device-width,
          initial-scale=1.0,
          minimum-scale=1.0,
          maximum-scale=1.0,
          user-scalable=false
        " />
      <?php } ?>
    </head>
    <body>
      <h1>Witaj, przeglądarko <?php print $device_group;?></h1>
...
```

Plik responsive2/Index.php

W tym fragmencie pominięto funkcje `device_property` i `get_http_header`, jednak ich kod pozostał taki sam. Ten program próbuje szybko ustalić, czy urządzenie ma ekran dotykowy i do której grupy określanej według szerokości wyświetlacza należy — szerokiej, średniej, wąskiej lub uproszczonej. Odpowiadają one przedziałom opisanym wcześniej w rozdziale. Warto zauważyć, że domyślna wielkość ekranu jest ustawiana na 160 pikseli dla urządzeń przenośnych i 640 pikseli dla pozostałych urządzeń, co jest przydatne, jeśli precyzyjne wartości są niedostępne w bazie danych. Można wtedy przynajmniej zapewnić płynną degradację.

Zgodnie z oczekiwaniami na rysunkach 9.22 i 9.23 interfejsy w różnych urządzeniach wyglądają inaczej.

Rysunek 9.22. Interfejs dla urządzeń z szerokim ekranem

Kiedy już wiadomo po stronie serwera, do której grupy należy dane urządzenie, można zmienić kod znaczników, co nie było możliwe przy stosowaniu samych zapytań Media Query. W poprzednim przykładzie technikę tę wykorzystano tylko do przełączania arkuszy stylów opartych na grupach (arkusze obejmują rozdzielone fragmenty z wcześniejszego pliku *narrow-medium-wide.css*). Można też usunąć deklarację `viewport` z urządzeń, co do których istnieje podejrzenie, że nie umożliwiają zmiany wielkości (za pomocą dotyku). Oczywiście są to uogólnienia, ponieważ naiwne jest założenie, że

Rysunek 9.23. Interfejs na wąskim wyświetlaczu

wszystkie urządzenia o ekranie o szerokości poniżej 241 pikseli są ograniczone. Jednak podejście to pokazuje, że można zastosować zarówno manipulowanie kodem znaczników, *jak i* dostosowywanie stylów po stronie klienta.

Można wykorzystać tę możliwość i zastanowić się nad różnymi priorytetami poszczególnych grup użytkowników. Jeśli tworzysz witrynę firmy, sklepu lub restauracji, możesz bardziej wyeksponować dane kontaktowe i mapę dojazdu w wersji mobilnej niż stacjonarnej (gdzie dane kontaktowe można umieścić w dolnej części strony).

Do pobrania w witrynie helion.pl

```
...
<body>
  <?php if (device_property($tree, $user_agent, 'mobileDevice', 0)) { ?>
    <div class='call_to_action'>
      Zadzwoń do nas na numer <a href='tel:555-1234-567'>555-1234-567</a>!
    </div>
  <?php } ?>
  <h1>Witaj, przeglądarko <?php print $device_group;?></h1>
  ...
```

Plik index.php

Zmiany tego rodzaju są dużo łatwiejsze do zaimplementowania, jeśli można zmodyfikować kod znaczników, zamiast tylko określać układ za pomocą wybiórczo stosowanych stylów CSS. Już samo to jest istotnym argumentem przeciw stosowaniu projektów opartych tylko na zapytaniach Media Query i stylach CSS.

Co ciekawe, reagowanie po stronie serwera zapewnia kilka dodatkowych możliwości w zakresie odpowiedzialnego udostępniania rysunków na urządzenia przenośne. Zamiast wysyłać ten sam rysunek na urządzenia w celu dostosowania jego wielkości do dowolnego rozmiaru ekranu, można wstępnie przygotować obrazki o odpowiedniej wielkości dla urządzeń z każdego przedziału.

```
<img src="lisa_<?php print $device_group; ?>.jpg" alt="The Mona Lisa"/>
```

Przy stosowaniu tego kodu trzeba udostępnić statyczne pliki o nazwach *lisa_narrow.jpg*, *lisa_medium.jpg* i tak dalej, jednak w ten sposób można utworzyć rysunki o mniej więcej odpowiedniej wielkości i uniknąć sytuacji, w której trzeba pobierać bardzo duże obrazki i zmniejszać je do małych

rozmiarów, co jest kosztowne. Inna możliwość to automatyczne zmienianie wielkości po stronie serwera, tak aby rysunki były idealnie dostosowane do każdego żądania, choć to rozwiązanie wymaga nieco więcej kodu. Jak zobaczysz, wiele wtyczek do systemów CMS udostępnia podobną funkcję.

Podsumowanie

Rozdział ten był stosunkowo teoretyczny, po części dlatego, że dziedzina projektów mobilnych dopiero się rozwija, ale też z uwagi na to, iż opracowanie dobrego projektu jest bardzo indywidualnym procesem i trudno dokładnie określić, które techniki będą odpowiednie przy uwzględnieniu Twoich umiejętności, systemu i witryny. Poznałeś jednak pewne podstawowe zasady projektowania interfejsów dostosowywanych zarówno po stronie serwera, jak i po stronie klienta. Dowiedziałeś się też, że warto próbować radzić sobie z różnorodnością urządzeń na wiele sposobów. Jeśli chodzi o estetyczne inspiracje w tym szybko rozwijającym się obszarze, możesz odwiedzić witrynę Mobile Awesomeness (*www.mobileawesomeness.com*), gdzie dostępne są galerie współczesnych projektów dla internetu mobilnego.

W następnym rozdziale poznasz szereg szablonów i bibliotek, które można wykorzystać do przyspieszenia podróży po świecie projektów mobilnych. Następnie przejdziesz do udostępniania materiałów z istniejących systemów CMS w internecie mobilnym.

Skorowidz

D

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

Ś

T

U

V

W

X

Z

Programowanie

mobilnych stron internetowych z wykorzystaniem systemów CMS

Świat oszalał na punkcie smartfonów, które wreszcie umożliwiły wygodne, szybkie przeglądanie Internetu i sprawiły, że korzystanie z zasobów sieci w telefonie stało się tak samo naturalną czynnością jak wykonywanie połączeń. To oczywiście zrodziło potrzebę tworzenia lżejszych, lecz tak samo funkcjonalnych oraz atrakcyjnych wizualnie odpowiedników różnych serwisów, e-sklepów i aplikacji, zapewniających komfortowe korzystanie z ich stron na wszelkich urządzeniach mobilnych.

Oto książka, w której znajdziesz szczegółowe omówienie technologii Internetu mobilnego — technik i narzędzi, które można wykorzystać do udostępnienia na urządzeniach przenośnych rozmaitych materiałów sieciowych. Przedstawiono tu standardowe wzorce rozwijania interfejsów użytkownika, a także ułatwiające pracę szablony i platformy. Autor koncentruje się na trzech najważniejszych CMS-ach (WordPress, Joomla! oraz Drupal) i wyjaśnia, jak projektować, budować oraz publikować za ich pomocą użyteczne witryny, które zachwycą użytkowników mobilnego Internetu!

W książce znajdziesz między innymi:

- **techniczne omówienie środowiska mobilnego oraz wprowadzenie do urządzeń i sieci mobilnych, a także związanych z nimi problemów;**
- **opis ewolucji Internetu mobilnego i optymalnych sposobów dostosowania się do niej;**
- **omówienie wyborów i decyzji, które trzeba podjąć przed udostępnieniem witryny opartej na systemie CMS w Internecie mobilnym;**
- **opis procesów związanych z testowaniem, instalowaniem i integrowaniem witryn mobilnych w dowolnym systemie;**
- **omówienie analizy, reklamy i innych zagadnień z obszaru mobilnego.**

Wyjdź naprzeciw możliwościom, jakie oferują Ci popularne platformy CMS, i twórz niezwykłe witryny z myślą o mobilnym Internecie.

James Pearce jest specjalistą od technologii, autorem książek, programistą i przedsiębiorcą pracującym w branży Internetu mobilnego od ponad dziesięciu lat. Jest także autorem popularnej wtyczki WordPress Mobile Pack.

helion.pl księgarnia internetowa

Nr katalogowy: 7848

Księgarnia internetowa: http://helion.pl

Zamówienia telefoniczne: 0 801 339900, 0 601 339900

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

sięgnij po WIĘCEJ — KOD KORZYŚCI

ISBN 978-83-246-3610-5

Cena 89,00 zł

Informatyka w najlepszym wydaniu