Projektowanie witryn internetowych dla urządzeń mobilnych

Autorzy: Gail Frederick, Rajesh Lal
Tłumaczenie: Mikołaj Szczepaniak
ISBN: 978-83-246-2729-5
Tytuł oryginału: Beginning Smartphone Web Development: Building Javascript, CSS, HTML and Ajax-Based Applications for iPhone, Android, Palm Pre, Blackberry, Windows Mobile and Nokia S60
Format: 158×235, stron: 376

Poznaj najlepsze praktyki tworzenia mobilnych witryn internetowych

- Jak budować interaktywne witryny dla smartfonów?
- Jak zwiększyć użyteczność mobilnej witryny internetowej?
- Jak zoptymalizować serwer WWW pod kątem przeglądarek mobilnych?

Internet mobilny to wyjątkowo ekscytujący, ale i chaotyczny ekosystem, który podlega gwałtownym i trudnym do przewidzenia zmianom. Ten podręcznik pozwoli Ci nie tylko przetrwać w tym fascynującym i „dziewiczym" świecie, ale skutecznie radzić sobie z pojawiającymi się w nim problemami – ze składnią, semantyką i programowaniem. Książka, którą trzymasz w rękach, to niezastąpiony przewodnik po dżungli mobilnego Internetu. Pozwoli Ci oswoić wszelkie problemy, z którymi możesz spotkać się jako programista.

Książka „Projektowanie witryn internetowych dla urządzeń mobilnych" zawiera szczegółowe opisy standardów, najlepszych praktyk i technik projektowania, niezbędnych do budowy interaktywnych stron WWW dla urządzeń mobilnych. Korzystając z tego podręcznika, nauczysz się stosować rozszerzenia dostępne dla zaawansowanych przeglądarek instalowanych w smartfonach, a także optymalizować, sprawdzać, testować oraz wdrażać witryny mobilne w publicznym Internecie i specyficznym ekosystemie mobilnym. Opanujesz niezbędne umiejętności i nabierzesz pewności siebie potrzebnej do tworzenia przenośnych aplikacji internetowych w niezwykłym środowisku urządzeń mobilnych.

- Mobilna witryna internetowa
- Mobilne arkusze stylów
- Konfiguracja środowiska
- Składnia mobilnych witryn internetowych
- Rozpoznawanie urządzeń i dostosowywanie treści
- Model wytwarzania iteracyjnego
- Wdrażanie rozwiązań w ekosystemie mobilnym
- Optymalizacja kodu mobilnego języka znaczników
- Testowanie i wdrażanie mobilnej witryny internetowej

Z tym podręcznikiem z łatwością oswoisz i uporządkujesz chaos ekosystemu Internetu mobilnego

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

Spis treści

ROZDZIAŁ

7

Wzbogacanie mobilnych stron internetowych z myślą o przeglądarkach instalowanych w smartfonach

Wszystkie techniki tworzenia mobilnych witryn internetowych omówione w pozostałych rozdziałach tej książki można z powodzeniem stosować w różnych urządzeniach i przeglądarkach mobilnych. Na podstawie bazy danych o urządzeniach i w drodze odpowiednich testów programista witryny mobilnej może urozmaicić swoje strony, dodając zaawansowane funkcje przeznaczone tylko dla najnowszych, najbardziej zaawansowanych przeglądarek. Tak rozbudowane strony można następnie dostosować do możliwości prostszych przeglądarek (instalowanych w tańszych telefonach). Warto przy tym zadbać o wybór właściwej wersji witryny, zależnej od używanego urządzenia mobilnego.

Rozdział ten poświęcono w całości przeglądarkom instalowanym w smartfonach. Możesz więc na moment zapomnieć o przenośności i zapewnianiu zgodności z wieloma platformami mobilnymi. Tym razem skoncentrujemy się na możliwościach najbardziej rozbudowanych przeglądarek mobilnych i najpopularniejszych obecnie smartfonów. Na tej podstawie opracujemy atrakcyjne aplikacje wykorzystujące potencjał tych zaawansowanych przeglądarek. Tego rozdziału nie należy jednak traktować jako wyczerpującej listy funkcji przeglądarek oferowanych przez poszczególne smartfony. Każda z tych przeglądarek zasługuje na osobną książkę! Tu dowiesz się, jakie są najważniejsze funkcje i ograniczenia poszczególnych przeglądarek oraz gdzie szukać podręczników dla programistów i dokumentacji producentów.

Wszystkie przeglądarki mobilne tutaj opisane obsługują język XHTML, standard CSS2 i skrypty języka JavaScript. Większość dodatkowo obsługuje technologię AJAX. Oznacza to, że przeglądarki instalowane w smartfonach stanowią efektywną platformę dla dynamicznych aplikacji internetu mobilnego. Naszą analizę rozpoczniemy od przeglądu typowych funkcji wspólnych dla wielu przeglądarek smartfonów. Zaraz potem przystąpimy do omawiania funkcji właściwych przeglądarkom

instalowanym w telefonach iPhone, w systemie Android, w systemie webOS, w telefonach BlackBerry, w telefonach Nokia Series 60 i w urządzeniach z systemem Windows Mobile. Zakończymy ten rozdział omówieniem przeglądarki Opera Mobile, czyli przeglądarki dla smartfonów opracowanej przez niezależnego producenta (także w zestawieniu z inną przeglądarką tego samego pochodzenia, czyli Operą Mini).

Popularne techniki tworzenia witryn dla przeglądarek instalowanych w smartfonach

Przeglądarki instalowane w smartfonach stosują takie same techniki przetwarzania i wyświetlania dokumentów języków znaczników. Przykładowo znacznik `<meta> viewport` kontroluje logiczne wymiary i skalowanie obszaru wyświetlania w przeglądarce. Zdarzenia `onresize` i `onorientationchange` języka JavaScript przechwytują zmiany orientacji ekranu urządzenia mobilnego (np. wskutek obrócenia z trybu poziomego do trybu pionowego). Właśnie JavaScriptu można użyć do określenia nowej orientacji ekranu po zmianie.

Oprócz dokumentacji technicznej udostępnianej przez producentów urządzeń i przeglądarek, warto zapoznać się także z doskonałymi tabelami obsługi zdarzeń i obiektów JavaScriptu przez urządzenia mobilne na blogu QuirksMode Petera-Paula Kocha pod adresem *www.quirksmode.org/m/table.html*.

Znacznik <meta> viewport

Wiele przeglądarek instalowanych w smartfonach skaluje strony internetowe, tak aby zajmowały całą szerokość obszaru wyświetlania. Takie rozwiązanie umożliwia prezentowanie dokumentów zoptymalizowanych pod kątem tradycyjnych przeglądarek internetowych. Przeglądarki umożliwiają użytkownikom przybliżanie i oddalanie skalowanych stron internetowych. Przykładowo przeglądarka Opera Mobile stosuje obszar wyświetlania o domyślnej szerokości 850 pikseli, natomiast w urządzeniach iPhone domyślna szerokość obejmuje 980 pikseli. Znacznik `<meta> viewport` umożliwia programiście mobilnej strony internetowej określenie najlepszego rozmiaru obszaru wyświetlania i — tym samym — wymuszenie skalowania stosowanych limitów dla danego dokumentu. Opisywany znacznik `<meta>` decyduje o logicznej szerokości, logicznej wysokości i początkowym współczynniku skalowania okna przeglądarki (lub obszaru wyświetlania) w przeglądarkach instalowanych w telefonach iPhone i BlackBerry oraz w pozostałych przeglądarkach mobilnych, w tym w przeglądarce Opera Mini. W niektórych smartfonach znacznik `<meta> viewport` dodatkowo określa, czy użytkownik może skalować daną stronę internetową i — jeśli tak — jakie są maksymalne i minimalne współczynniki skalowania. Sama obecność znacznika `<meta> viewport` określa, że dany dokument języka znaczników zoptymalizowano z myślą o urządzeniach mobilnych.

Wartością atrybutu `content` znacznika `<meta> viewport` powinna być lista oddzielonych przecinkami par dyrektywa-wartość. Poniżej przedstawiono znacznik `<meta>` wymieniający wszystkie te dyrektywy wraz z przykładowymi wartościami:

```
<meta name="viewport" content="width=240, height=320, user-scalable=yes,
  initial-scale=2.5, maximum-scale=5.0, minimum-scale=1.0" />
```

Dyrektywy `width` i `height` określają odpowiednio logiczną szerokość i wysokość obszaru wyświetlania. Dyrektywom tym należy przypisać albo wartość numeryczną reprezentującą szerokość obszaru wyświetlania (w pikselach), albo specjalny token. Dyrektywie `width` można

przypisać token `device-width`, który określa, że szerokość obszaru wyświetlania powinna odpowiadać szerokości ekranu danego urządzenia. Podobnie dyrektywie `height` można przypisać token `device-height` określający, że wysokość obszaru wyświetlania powinna odpowiadać wysokości ekranu danego urządzenia

Dyrektywa `user-scalable` ustala, czy dany użytkownik może przybliżać i oddalać obszar wyświetlania, skalując widok strony internetowej. Wartość `yes` umożliwia użytkownikom przybliżanie strony, natomiast wartość `no` zapobiega przybliżaniu i skalowaniu strony przez użytkownika. Programista mobilnej witryny internetowej może zapobiec możliwości skalowania stron przez użytkownika, przypisując tej dyrektywie wartość `no` — takie rozwiązanie jest uzasadnione dla stron zoptymalizowanych pod kątem urządzeń mobilnych, które nie wymagają przybliżania i oddalania obrazów.

Dyrektywa `initial-scale` ustawia początkową wartość współczynnika, tzw. mnożnika skalowania (przybliżenia) dla danej strony internetowej. Domyślna wartość początkowa zależy od przeglądarki stosowanej w smartfonie. W większości przypadków wartość początkową ustawia się w taki sposób, aby cała strona mieściła się w obszarze wyświetlania. Wartość `1.0` powoduje wyświetlenie dokumentu bez skalowania.

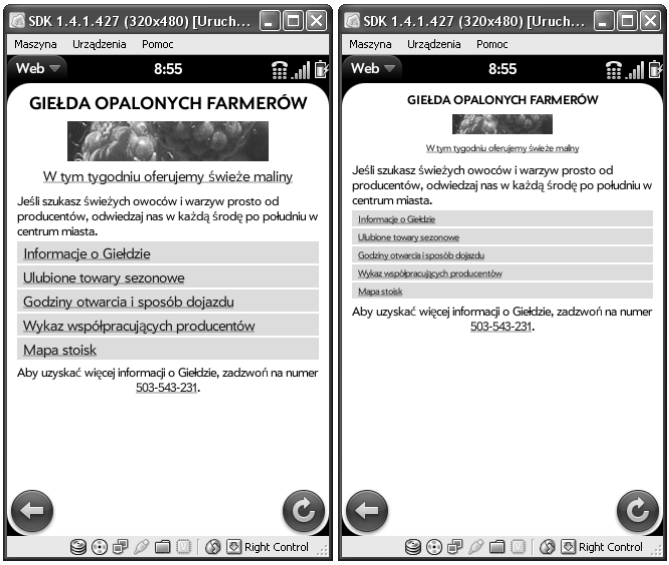
Dyrektywy `maximum-scale` i `minimum-scale` określają odpowiednio maksymalną i minimalną wartość współczynnika skalowania (przybliżania) ustawianego przez użytkownika. Wartości przypisywane tym dyrektywom muszą mieścić się w przedziale od `0.25` do `10.0`. Podobnie jak w przypadku dyrektywy `initial-scale`, wartości tych dyrektyw dotyczą współczynnika (mnożników) skalowania stosowanego dla treści w obszarze wyświetlania.

Niemal wszystkie przeglądarki instalowane w smartfonach obsługują dyrektywy `width` i `user-scalable` znacznika `<meta>` `viewport`. Warto zwrócić uwagę na brak obsługi dyrektywy `user-scalable` przez przeglądarkę mobilną Opera Mobile — twórcy tej przeglądarki uznali, że użytkownicy mobilni zawsze powinni mieć możliwość skalowania stron internetowych. Wiele przeglądarek instalowanych w smartfonach i korzystających z silnika WebKit (więcej informacji na ten temat można znaleźć w podrozdziale „Silnik WebKit w mobilnych przeglądarkach internetowych") obsługuje wszystkie dyrektywy znacznika `<meta>` nazwanego `viewport`. Ponieważ w znaczniku `<meta>` programista deklaruje rozmiar i sposób skalowania obszaru wyświetlania, przeglądarki, które nie obsługują tych dyrektyw, powinny bezpiecznie i bez angażowania użytkownika (np. poprzez wyświetlanie komunikatów o błędach) ignorować odpowiednie konstrukcje.

Poniższy przykład znacznika `<meta>` wymusza na przeglądarce mobilnej ustawienie szerokości obszaru wyświetlania równej szerokości ekranu smartfonu (niezależnie od wartości tego parametru) oraz wyłączenie możliwości skalowania dokumentu przez użytkownika. Znacznik `<meta>` w tej formie jest dość często stosowany w kodzie stron optymalizowanych pod kątem przeglądarek mobilnych, ponieważ umożliwia użytkownikowi przeglądanie i przewijanie treści strony bez możliwości jej przybliżania i oddalania:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0,
  user-scalable=no" />
```

Na rysunku 7.1 pokazano przykład mobilnej strony internetowej z listingu 9.2 (z rozdziału 9.) w wersjach z powyższym znacznikiem `<meta>` `viewport` oraz bez tego znacznika wyświetlonych na ekranie smartfonu Palm Pre. Warto zwrócić uwagę na sposób, w jaki znacznik `<meta>` `viewport` wpływa na skalowanie i początkową użyteczność tej strony. Wyłączenie możliwości przewijania i ustawienie odpowiedniej szerokości obszaru wyświetlania w praktyce wyklucza konieczność przybliżania widoku treści strony.



Rysunek 7.1. Mobilne strony internetowe odpowiednio ze znacznikiem <meta> viewport i bez tego znacznika wyświetlone na ekranie telefonu Palm Pre

Wykrywanie zmian orientacji ekranu w JavaScripcie

Wiele smartfonów aktualizuje orientację ekranu w odpowiedzi na zmianę fizycznej orientacji urządzenia przez użytkownika. Jeśli np. użytkownik obraca smartfon z orientacji poziomej do orientacji pionowej, urządzenie reaguje odpowiednią zmianą orientacji ekranu. Przeglądarki smartfonów informują o tym fakcie, korzystając ze zdarzeń `onresize` lub `onorientationchange` obiektu `window` JavaScriptu (na podstawie tych zdarzeń programista mobilnej witryny internetowej może właściwie reagować na zmiany orientacji).

Przeglądarki instalowane w smartfonach obsługują albo jedno z tych zdarzeń, albo oba zdarzenia, zatem niezwykle ważne jest przeprowadzenie testów witryny w samych urządzeniach, aby wybrać właściwe zdarzenie **do obsłużenia** w docelowej przeglądarce mobilnej. Przeglądarka telefonów iPhone obsługuje oba zdarzenia, jednak większość programistów mobilnych witryn internetowych decyduje się na wykrywanie zmian orientacji tylko na podstawie zdarzenia `onorientationchange`. Urządzenia BlackBerry obsługują, co prawda, zdarzenie `onresize` obiektu `document`, jednak niewiele spośród tych urządzeń oferuje możliwość zmiany orientacji ekranu.

W kodzie funkcji obsługującej zdarzenie zmiany orientacji programista może użyć jednej z dwóch metod do uzyskania bieżących wymiarów ekranu i bieżącej orientacji urządzenia. Programiści witryny dla urządzeń iPhone mają do dyspozycji wbudowaną właściwość całkowitoliczbową JavaScriptu nazwaną `window.orientation`. Wartość tej właściwości reprezentuje bieżącą orientację przeglądarki. W tabeli 7.1 opisano możliwe wartości właściwości `window.orientation` wraz z ich znaczeniem.

W poniższym przykładzie kodu języka JavaScript wykorzystano wbudowaną właściwość `window.orientation` do określenia, czy dany smartfon jest trzymany w orientacji poziomej, czy pionowej.

Tabela 7.1. Wartości właściwości window.orientation języka JavaScript dla telefonu iPhone

Wartość właściwości	Opis
-90	Urządzenie pracuje w trybie poziomym, gdzie ekran obrócono zgodnie z kierunkiem ruchu wskazówek zegara (przyciski urządzenia znajdują się po lewej stronie).
0	Urządzenie pracuje w trybie pionowym (0 jest domyślną wartością tej właściwości).
90	Urządzenie pracuje w trybie poziomym, gdzie ekran obrócono w kierunku przeciwnym do ruchu wskazówek zegara (przyciski urządzenia znajdują się po prawej stronie).
180	Urządzenie pracuje w trybie pionowym, jednak ekran obrócono do góry nogami. (Telefony iPhone na razie nie obsługują tej opcji, jednak niewykluczone, że przyszłe aktualizacje firmware'u wprowadzą odpowiedni mechanizm).

```
switch (window.orientation) {
    case -90:
        // Orientacja pozioma po obróceniu zgodnie z kierunkiem ruchu wskazówek zegara.
        break;
    case 0:
        // Orientacja pionowa.
        break;
    case 90:
        // Orientacja pozioma po obróceniu w kierunku przeciwnym do ruchu wskazówek zegara.
        break;
}
```

Możesz też wykorzystać wbudowane właściwości `screen.width` i `screen.height` języka JavaScript i określić orientację ekranu na podstawie prostych obliczeń matematycznych. Jeśli szerokość ekranu jest większa od jego wysokości, mamy do czynienia z orientacją poziomą. W przeciwnym razie urządzenie działa w orientacji pionowej. W poniższym fragmencie kodu JavaScriptu określono orientację urządzenia właśnie na podstawie wbudowanych właściwości JavaScriptu:

```
var width = parseInt(screen.width);
var height = parseInt(screen.height);

if (width > height) {
    // Orientacja pozioma.
}
else {
    // Orientacja pionowa.
}
```

W powyższym przykładzie użyliśmy funkcji wbudowanej `parseInt()` do zapewnienia, że wartości reprezentujące szerokość i wysokość ekranu są liczbami całkowitymi — w ten sposób unikamy błędów w implementacjach JavaScriptu, w których obie te właściwości mają postać łańcuchów.

W niektórych przypadkach przeglądarka mobilna (inna niż przeglądarki instalowane w smartfonach) może nie zaktualizować wartości właściwości `screen.width` i `screen.height` w reakcji na zmianę orientacji urządzenia. Przeciwnie, wymienione właściwości zachowują wartości dla orientacji pionowej. Podobne zachowanie nie występuje, co prawda, w smartfonach, jednak zdarza się w pełnowartościowych przeglądarkach internetowych instalowanych w popularnych urządzeniach mobilnych. W tej sytuacji niezwykle ważne jest przeprowadzenie testów samych urządzeń, aby wyeliminować wpływ tego błędu na działanie naszej aplikacji internetu mobilnego.

W listingu 7.1 przedstawiono przykład dokumentu mobilnej witryny internetowej zgodnego z urządzeniem iPhone. Strona w tej formie obsługuje zdarzenie `onorientationchange` i na jego podstawie wykrywa zmiany orientacji urządzenia. Funkcja obsługująca to zdarzenie przekazuje informacje o bieżącej orientacji za pośrednictwem właściwości `window.orientation`.

Listing 7.1. Obsługa zmian orientacji za pomocą zdarzenia onorientationchange

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.1//EN"
"http://www.openmobilealliance.org/tech/DTD/xhtml-mobile11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="viewport" content="width=device-width,user-scalable=no" />
<title>Listing 7.1: Zmiany orientacji reprezentowane przez onorientationchange</title>
<style type="text/css">
#content {
    padding: 10px;
    margin: 10px;
    border-width: 1px;
    border-style: solid;
    border-color: #333333;
}
</style>
<script type="text/javascript">

// Przechwytuje zdarzenie zmiany orientacji:
function handleOrientationChange() {
    // Określa bieżącą orientację:
    var orientation = "unknown";
    switch (window.orientation){
        case -90:
            orientation = "Pozioma (zgodnie z kierunkiem ruchu wskazówek zegara)";
            break;
        case 0:
            orientation = "Pionowa";
            break;
        case 90:
            orientation = "Pozioma (w kierunku przeciwnym do ruchu wskazówek zegara)";
            break;
    }

    // Dodaje bieżące wymiary ekranu:
    var screenSize = screen.width + " x " + screen.height;

    // Aktualizuje informacje o orientacji dla użytkownika:
    document.getElementById("content").innerHTML = orientation + ", "
        + screenSize;
```

```
}
</script>
</head>
<!-- W tym przypadku generujemy zdarzenie onorientationchange także przy okazji ładowania dokumentu,
   aby wyświetlić na ekranie informacje o domyślnej orientacji urządzenia mobilnego. -->
<body onload="handleOrientationChange();"
onorientationchange="handleOrientationChange();">
<h1>Zmiany orientacji</h1>
<p>Zmień orientację swojego urządzenia mobilnego. Sprawdź w poniższym polu,
jaka jest bieżąca orientacja tego urządzenia.</p>
<div id="content"></div>
</body>
</html>
```

W listingu 7.1 funkcja `handleOrientationChange` języka JavaScript odpowiada za obsługę zdarzenia `onorientationchange`. Funkcja oblicza orientację na podstawie właściwości `window.orientation`. Użytkownik jest informowany o bieżącej orientacji w wyniku aktualizacji właściwości `innerHTML` elementu `div` z identyfikatorem `content`. Warto otworzyć dokument z listingu 7.1 w przeglądarce zainstalowanej w smartfonie, aby sprawdzić, jak zmiany orientacji tego urządzenia wpływają na treść strony.

Na rysunku 7.2 pokazano dokument z listingu 7.1 wyświetlony w emulatorze smartfonu iPhone zarówno w orientacji pionowej, jak i poziomej. Warto zwrócić uwagę na tekst w ramach elementu `div`, któremu przypisano identyfikator `content`. Okazuje się, że do określania orientacji ekranu nie musimy używać jego wymiarów. Ponieważ iPhone obsługuje właściwość `window.orientation`, zatem orientację urządzenia możemy określić bez odwoływania się do wartości właściwości `screen.width` i `screen.height`.

Rysunek 7.2. Dokument z listingu 7.1 wyświetlony w emulatorze telefonu iPhone w orientacji pionowej i poziomej

W listingu 7.2 przedstawiono przykład dokumentu mobilnej witryny internetowej wykrywającego i obsługującego zmiany orientacji urządzenia mobilnego na podstawie zdarzenia `onresize`. Funkcja obsługująca to zdarzenie wykorzystuje wartości właściwości `screen.width` i `screen.height` do wyznaczania i wyświetlania bieżącej orientacji urządzenia.

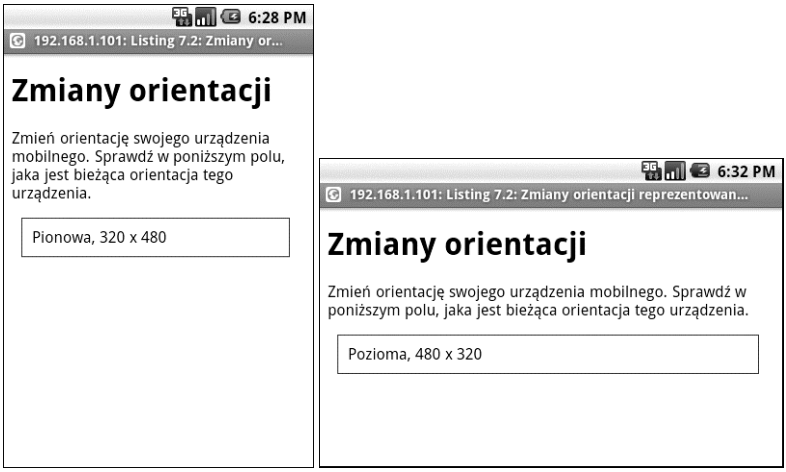
Listing 7.2. Obsługa zmian orientacji za pomocą zdarzenia onresize

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.1//EN"
  "http://www.openmobilealliance.org/tech/DTD/xhtml-mobile11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="viewport" content="width=device-width,user-scalable=no" />
<title>Listing 7.2: Zmiany orientacji reprezentowane przez onresize</title>
<style type="text/css">
#content {
    padding: 10px;
    margin: 10px;
    border-width: 1px;
    border-style: solid;
    border-color: #333333;
}
</style>
<script type="text/javascript">
// Przechwytuje zdarzenie zmiany orientacji:
function handleResize() {

    // Określa bieżącą orientację:
    var orientation = "unknown";
    var width = parseInt(screen.width);
    var height = parseInt(screen.height);

    if (width > height) {
        orientation = "Landscape";
    }
    else {
        orientation = "Portrait";
    }

    // Dodaje bieżące wymiary ekranu:
    var screenSize = screen.width + " x " + screen.height;

    // Aktualizuje informacje o orientacji dla użytkownika:
    document.getElementById("content").innerHTML = orientation + ", "
        + screenSize;
}
</script>
</head>
<!-- W tym przypadku generujemy zdarzenie onresize także przy okazji ładowania dokumentu,
   aby wyświetlić na ekranie informacje o domyślnej orientacji urządzenia mobilnego. -->
<body onload="handleResize();" onresize="handleResize();">
<h1>Zmiany orientacji</h1>
<p>Zmień orientację swojego urządzenia mobilnego. Sprawdź w poniższym polu,
jaka jest bieżąca orientacja tego urządzenia.</p>
<div id="content"></div>
</body>
</html>
```

Funkcja `handleResize` języka JavaScript w kodzie z listingu 7.2 odpowiada za obsługę zdarzenia `onresize`. Funkcja ta wyznacza orientację na podstawie wartości właściwości `screen.width` i `screen.height`. Podobnie jak w przypadku dokumentu z listingu 7.1, o bieżącej orientacji

urządzenia informujemy użytkownika, aktualizując element właściwości `innerHTML` elementu `div` z identyfikatorem `content`. Warto otworzyć dokument z listingu 7.2 w przeglądarce zainstalowanej w smartfonie, aby sprawdzić, jak zmiany orientacji tego urządzenia wpływają na treść strony.

Na rysunku 7.3 pokazano dokument z listingu 7.2 wyświetlony w emulatorze systemu Android zarówno w orientacji pionowej, jak i w orientacji poziomej. Warto zwrócić uwagę na zmieniającą się zawartość elementu `div`, któremu przypisano identyfikator `content`. Tym razem orientację urządzenia określamy na podstawie wartości właściwości `screen.width` i `screen.height`.



Rysunek 7.3. Dokument z listingu 7.2 wyświetlony w emulatorze systemu Android w orientacji pionowej i poziomej

Silnik WebKit w mobilnych przeglądarkach internetowych

WebKit to silnik zarządzania układem i wyświetlania dokumentów wykorzystywany w przeglądarkach internetowych. WebKit stosuje się w przeglądarkach, w których zgodność z rygorystycznymi standardami jest ważniejsza od wydajności procesu wizualizacji stron. Silnik WebKit w oryginalnej formie zaimplementowano w języku C++, jednak z czasem przeniesiono go do wielu różnych frameworków programistycznych stosowanych zarówno w tradycyjnych środowiskach biurkowych, jak i w środowiskach mobilnych.

Firma Apple opracowała własny, oryginalny silnik WebKit dla tradycyjnej przeglądarki internetowej Safari w wersji dla systemu Mac OS X. W tym wydaniu silnika WebKit wykorzystano i usprawniono komponenty języków HTML i JavaScript zaczerpnięte z projektu open-source KDE. W 2005 roku firma Apple udostępniła ten framework w trybie open-source i dopuściła programistów do zasobu systemu kontroli wersji tego projektu. Od tamtego czasu za rozwój silnika WebKit odpowiada WebKit Open Source Project (*http://webkit.org/*). We wspomnianym projekcie może uczestniczyć każdy zainteresowany programista (także Ty!). Warto odwiedzić stronę *http://trac.webkit.org/timeline*, gdzie na bieżąco są umieszczane wpisy o zmianach w kodzie źródłowym. Programiści zaangażowani w prace nad przeglądarkami internetowymi takich firm jak Apple, Google, Nokia czy Palm niemal codziennie wprowadzają cenne poprawki w kodzie silnika WebKit.

Silnik przeglądarek WebKit składa się z dwóch głównych bibliotek: WebCore i JavaScript Core. Obie biblioteki łącznie kontrolują procesy wizualizowania strony internetowej i zarządzania funkcjonowaniem elementów interaktywnych po stronie klienta. Silnik WebKit obsługuje wymienione niżej standardy internetowe.

- **Znaczniki:** HTML 4.01, XHTML 1.0 oraz pewne elementy standardu HTML 5.
- **Style:** CSS 2.1 i pewne elementy standardu CSS3.
- **Skrypty:** JavaScript 1.8, AJAX oraz DOM Level 1, 2 i 3.

Z silnika wyświetlania dokumentów WebKit korzystają tradycyjne przeglądarki internetowe Apple Safari i Google Chrome. Ten sam silnik cieszy się także sporą popularnością wśród twórców przeglądarek opracowywanych z myślą o smartfonach przede wszystkim ze względu na obsługę wymienionych powyżej języków znaczników, standardów arkuszy stylów i języków skryptowych. Na bazie technologii WebKit zaimplementowano przeglądarki dla takich systemów jak iPhone, Android, webOS czy Nokia Series 60.

Po lekturze wielu rozdziałów, w których nieustannie wskazywano na pogłębiające się różnice dzielące poszczególne urządzenia i przeglądarki mobilne, możesz wreszcie rozkoszować się perspektywą powstania jednej, spójnej implementacji silnika WebKit w przeglądarkach instalowanych w smartfonach. Okazuje się jednak, że producenci mobilnych przeglądarek internetowych mogą korzystać z silnika wyświetlania dokumentów WebKit i jednocześnie wprowadzać w tym silniku zmiany gwarantujące odpowiednią wydajność w urządzeniu mobilnym z ograniczonymi zasobami (smartfony wciąż nie dorównują mocą obliczeniową tradycyjnym komputerom). Właśnie dlatego implementacje silnika WebKit występują w różnych odmianach. Peter-Paul Koch, autor popularnego bloga QuirksMode (*www.quirksmode.org*), przetestował 19 implementacji silnika WebKit (w tym 10 implementacji dla przeglądarek instalowanych w smartfonach) i odkrył nieznaczne różnice dzielące zakres obsługiwanych elementów standardów CSS i JavaScript. Opracowaną przez Kocha tabelę porównawczą tych implementacji można znaleźć na stronie *http://www.quirksmode.org/webkit.html*. Wniosek? Wiele przeglądarek instalowanych w smartfonach korzysta z silnika WebKit, zapewniając programistom witryn mobilnych niemal w pełni spójną platformę dla bogatych aplikacji internetowych. Jak zawsze w takich przypadkach, należy testować tworzone aplikacje w samych urządzeniach mobilnych, aby w ten sposób weryfikować zgodność na poziomie składni i wydajność wyświetlania dokumentów (więcej informacji na temat mobilnych aplikacji internetowych można znaleźć w rozdziale 10.).

Silnik WebKit wprowadza wiele rozszerzeń standardu CSS, aby umożliwić stosowanie w oknie przeglądarki stylów z zaawansowanymi efektami wizualnymi. Część właściwości arkuszy stylów CSS w silniku WebKit wyprzedza rozwiązania, które zostaną wprowadzone dopiero w nadchodzących wydaniach tego standardu. W tabeli 7.2 podsumowano kilka najczęściej używanych rozszerzeń standardu CSS w ramach silnika WebKit. Nie wszystkie te rozszerzenia są implementowane we wszystkich przeglądarkach mobilnych z silnikiem WebKit, ale wszystkie rozszerzenia wymienione i opisane w tabeli 7.2 są obsługiwane przez przeglądarkę Safari Mobile instalowaną w urządzeniu iPhone.

W listingu 3.7 podajemy przykład dokumentu mobilnej witryny internetowej wykorzystującego kilka rozszerzeń standardu CSS opisanych w tabeli 7.2. Warto otworzyć ten dokument w jednej z mobilnych przeglądarek internetowych zbudowanych na bazie silnika WebKit.

Tabela 7.2. Wybrane rozszerzenia standardu CSS w silniku WebKit

Nazwa właściwości	Wartość właściwości	Opis
`-webkit-background-size`	Jedna lub dwie wartości całkowitoliczbowe (w pikselach)	Ustawia rozmiar obrazu tła.
`-webkit-border-radius`	Wartość całkowitoliczbowa (w pikselach)	Określa promień zaokrąglonego narożnika elementu prostokąta.
`-webkit-box-shadow`	Dwie szerokości cienia (w pikselach) oraz wartość koloru	Określa rozmiar cienia rzucanego przez dany obraz.
`-webkit-transform`	Jedna z wielu funkcji przekształceń arkusza stylów CSS	Stosuje przekształcenie wizualne dla danego elementu (polegające na zmianie skali, obrocie, przesunięciu itp.).
`-webkit-text-sizeadjust`	Wartość procentowa	Zmienia skalę tekstu o podaną wartość procentową, aby zwiększyć lub zmniejszyć rozmiar tekstu.

Listing 7.3. Dokument mobilnej witryny internetowej korzystający z rozszerzeń standardu CSS obsługiwanych przez silnik WebKit

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.1//EN"
"http://www.openmobilealliance.org/tech/DTD/xhtml-mobile11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="viewport" content="width=device-width,user-scalable=no" />
<title>Listing 7.3: Rozszerzenia CSS silnika WebKit</title>
<style type="text/css">
#content {
    width: 89px;
    height: 89px;
    padding: 10px;
    margin: 10px;
    border: 2px solid #333;
    -webkit-border-radius: 5px;
    -webkit-transform: rotate(15deg);
}
#tulips {
    width: 79px;
    height: 79px;
    margin: 5px;
}
</style>
</head>
<body>
<h1>Rozszerzenia CSS silnika WebKit</h1>
<div id="content"><img id="tulips" src="tulips.jpg" /></div>
</body>
</html>
```

Na rysunku 7.4 pokazano zrzut ekranu ilustrujący dokument mobilnej witryny internetowej z listingu 7.3 wyświetlony w emulatorach urządzenia Palm Pre i systemu Android. Warto zwrócić uwagę na sposób, w jaki rozszerzenia standardu CSS oferowane przez silnik WebKit wzbogacają doznania użytkowników mobilnych korzystających ze smartfonów (w tym przypadku poprzez zaokrąglenie narożników obramowania elementu `div` i obrócenie obrazu tulipana).

Rysunek 7.4. Dokument z listingu 7.3 otwarty w emulatorach urządzenia Palm Pre i systemu Android

Na rysunku 7.5 pokazano dokument mobilnej witryny internetowej z listingu 7.3 wyświetlony w małym oknie przeglądarki Opera Mobile, która nie obsługuje rozszerzeń CSS silnika WebKit. Jak widać, nasza strona internetowa prawidłowo dostosowuje się do tych warunków. Ponieważ rozszerzenia właściwe tylko silnikowi WebKit wyodrębniono w arkuszu stylów, przeglądarka mobilna po prostu ignoruje nieznane style.

Rysunek 7.5. Dokument z listingu 7.3 otwarty w małym oknie przeglądarki Opera Mobile

Przeglądarka Safari Mobile dla urządzenia iPhone

Domyślną przeglądarką internetową instalowaną w telefonach iPhone jest Safari Mobile. Firma Apple słynie z dość niechętnego wdrażania standardów internetu mobilnego w oprogramowaniu swojego smartfonu i przekonuje, że jej przeglądarka (w przeciwieństwie do produktów konkurencji) obsługuje wyłącznie standardy znane z tradycyjnego, biurkowego internetu. Przeglądarka Safari Mobile rzeczywiście nie wyświetla dokumentów opracowanych w starszym języku znaczników WML, ale doskonale radzi sobie z prezentowaniem witryn mobilnych napisanych przy użyciu nowszych języków. To o tyle naturalne, że przeglądarka Safari Mobile korzysta z tego samego silnika WebKit, który wykorzystano w implementacji biurkowej wersji przeglądarki Safari firmy Apple.

Serwis iPhone Dev Center (*http://developer.apple.com/iphone/*) zawiera mnóstwo informacji na temat standardów internetowych, najlepszych praktyk i technik wytwarzania mobilnych witryn internetowych. Pełny dostęp do dokumentacji programistycznej oferowanej w tym serwisie wymaga od użytkownika darmowej rejestracji. Właściciele komputerów z systemem Mac OS X mogą dodatkowo pobrać i zainstalować pakiet iPhone SDK wraz z emulatorem, który symuluje działanie przeglądarki Safari Mobile. Bodaj najcenniejszym obszarem serwisu iPhone Dev Center jest sekcja dokumentacji programistycznej, która w założeniu ma ułatwiać programistom lepsze rozumienie struktury, elementów użyteczności i technologii zastosowanych w smartfonie iPhone.

Z perspektywy programistów mobilnych witryn internetowych dla telefonu iPhone zdecydowanie najważniejszym dokumentem jest Safari Web Content Guide (*http://developer.apple.com/safari/library/documentation/AppleApplications/Reference/SafariWebContent/Introduction/Introduction.html*); dokument jest dostępny w sekcji iPhone OS Reference Library serwisu iPhone Dev Center. Wspomniany dokument uczy, jak projektować i tworzyć witryny internetowe pod kątem telefonów iPhone i środowiska przeglądarki instalowanego w tych urządzeniach. Z dokumentu Safari Web Content Guide można się dowiedzieć np. tego, że w telefonach iPhone instaluje się pamięć w ilości wystarczającej do obsługi następujących rozmiarów zasobów internetowych.

- Obrazy w formatach GIF, PNG i TIFF mogą zajmować maksymalnie po 3 MB.
- Obrazy w formacie JPEG mogą zajmować maksymalnie po 32 MB.
- Pliki języków JavaScript i HTML oraz arkuszy stylów CSS nie mogą być większe niż 10 MB!
- Skrypt języka JavaScript może być wykonywany przez maksymalnie dziesięć sekund (po tym czasie jest przerywany).
- Przeglądarka Safari Mobile może jednocześnie utrzymywać maksymalnie osiem otwartych dokumentów internetowych.

Uwaga! Mimo że na pierwszy rzut oka rozmiary zasobów internetowych obsługiwanych przez samo urządzenie iPhone wydają się niemal nieograniczone, prawdziwym źródłem ograniczeń i utrudnień w przypadku treści tak dużych rozmiarów mogą być sieci mobilne. Więcej informacji na temat przyczyn, dla których próby udostępniania megabajtów tekstu i danych binarnych na witrynach kierowanych do właścicieli smartfonów są nierealistyczne (lub wręcz niegrzeczne względem samych użytkowników), można znaleźć w rozdziałach 8., 11. i 12. oraz w dodatku D.

Odtwarzacz iPod Touch firmy Apple jest niemal identyczny z telefonem iPhone z tą różnicą, że nie oferuje możliwości uzyskiwania dostępu do mobilnej sieci radiowej. Przeglądarka Safari Mobile działa tak samo w telefonie iPhone i w odtwarzaczu iPod Touch. Oznacza to, że wszystkie witryny internetowe zoptymalizowane z myślą o telefonach iPhone są optymalizowane także dla odtwarzaczy iPod Touch, jeśli tylko serwer właściwie rozpoznaje identyfikator aplikacji klienckiej tego odtwarzacza.

Na rysunku 7.6 pokazano zrzut ekranu przeglądarki internetowej Safari Mobile w emulatorze iPhone Simulator (jednym z komponentów pakietu iPhone SDK).

Rysunek 7.6. Przeglądarka Safari Mobile w emulatorze telefonu iPhone

Przeglądarka dla urządzeń mobilnych z systemem Android

Przeglądarka internetowa instalowana w urządzeniach mobilnych z systemem Android korzysta z silnika WebKit w wersji opracowanej przez firmę Apple. (Silnik WebKit szczegółowo opisano w podrozdziale „Silnik WebKit w mobilnych przeglądarkach internetowych", we wcześniejszej części tego rozdziału). Implementacja silnika WebKit w smartfonach z systemem Android obejmuje pełną obsługę standardów internetowych oraz pewnych elementów języka HTML 5, takich jak lokalne bazy danych (niezbędne do przeglądania lokalnego). Wszystkie wersje tej przeglądarki obsługują oprogramowanie Google Gears.

Okazuje się, że oprócz dokumentacji silnika WebKit dostępnej na witrynie *http://webkit.org* istnieje jeszcze wiele oficjalnych dokumentów opisujących funkcje przeglądarek instalowanych w poszczególnych wersjach systemu Android OS. Programiści muszą sami identyfikować wersje silnika WebKit na podstawie wartości nagłówka `User-Agent` i na tej podstawie określać parametry przeglądarek. Część dokumentacji przeglądarki systemu Android można znaleźć na oficjalnej witrynie internetowej Android Developers (*http://developer.android.com/index.html*). Witrynę opracowano, co prawda, z myślą o programistach budujących aplikacje specjalnie dla urządzeń z systemem Android, jednak próżno szukać na niej wielu ważnych informacji o samych przeglądarkach. Programiści aplikacji internetowych powinni pobrać i zainstalować darmowy pakiet narzędzi programistycznych Android SDK, aby emulować działanie urządzeń i przeglądarki.

Jednym z najbardziej przydatnych narzędzi programistycznych dostępnych w pakiecie Android SDK jest debugger mobilnych aplikacji internetowych. Za jego pomocą można skutecznie identyfikować problemy związane ze skryptami JavaScriptu i elementami technologii AJAX, których odtwarzanie i odnajdywanie w emulatorze byłoby trudne.

Aby diagnozować kod JavaScriptu wykonywany w przeglądarce w emulatorze lub w samym systemie Android (w urządzeniu mobilnym), zainstaluj pakiet Android SDK i włącz narzędzie Android Debug Bridge (*http://developer.android.com/guide/developing/tools/adb.html*). Narzędzie Android Debug Bridge łączy pakiet Android SDK z działającym emulatorem lub urządzeniem mobilnym. Po ustanowieniu tego połączenia uzyskujesz dostęp do zapisów diagnostycznych i komunikatów o błędach generowanych przez znacznik `WebCore`. Kierowanie komunikatów o błędach i informacji diagnostycznych JavaScriptu do okna konsoli wymaga aktywacji pakietu Android SDK.

Na rysunku 7.7 pokazano wyniki wygenerowane przez wyszukiwarkę Google w przeglądarce systemu Android w emulatorze urządzenia Android G1 (wchodzącym w skład pakietu Android SDK).

Rysunek 7.7. Przeglądarka systemu Android w emulatorze urządzenia Android G1

Przeglądarka systemu webOS dla urządzenia Palm Pre

Przeglądarka systemu webOS w smartfonach Palm Pre (i Pixi) korzysta z silnika WebKit gwarantującego obsługę wielu różnych standardów internetowych (patrz podrozdział „Silnik WebKit w mobilnych przeglądarkach internetowych", we wcześniejszej części tego rozdziału). Przeglądarka systemu webOS obsługuje wybrane elementy języka HTML 5, w tym część funkcji elementu `canvas` oraz lokalne bazy danych (czyli technologię umożliwiającą przeglądanie mobilnych aplikacji internetowych w trybie offline). W praktyce przeglądarka systemu webOS bardzo przypomina przeglądarkę Safari Mobile.

Społeczność programistów oprogramowania dla systemu webOS (*http://developer.palm.com/*) jest otwarta dla każdego zainteresowanego — wystarczy darmowa rejestracja. Z witryny tej społeczności można pobrać emulatory systemu webOS i pakiet narzędzi webOS SDK. Na tej samej witrynie można też znaleźć dokumentację i blogi programistów poświęcone systemowi webOS, aplikacjom wstępnie instalowanym w urządzeniach Palm (w tym przeglądarce tego sytemu i wskazówkom dotyczącym programowania) oraz technikom i najlepszym praktykom programowania. Podczas tworzenia aplikacji dla platformy webOS zdecydowana większość

programistów mobilnych stosuje standardy technologii internetowych. Oznacza to, że Twoja znajomość języków XHTML i JavaScript oraz standardu CSS powinna wystarczyć jako podstawa do bliższego poznawania technik budowy bogatych aplikacji dla urządzeń z systemem webOS.

Na rysunku 7.8 pokazano wyniki wygenerowane przez wyszukiwarkę Google w przeglądarce systemu webOS (w ramach emulatora urządzenia Palm Pre).

Rysunek 7.8. Przeglądarka systemu webOS w emulatorze urządzenia Palm Pre

Przeglądarka dla urządzeń BlackBerry

Przeglądarka BlackBerry Browser jest instalowana we wszystkich smartfonach BlackBerry. W przeszłości przeglądarka ta obsługiwała tylko standardy internetu mobilnego (XHTML-MP, WML i Wireless CSS) i słynęła ze stosunkowo niskiej wydajności oraz niewielkiej innowacyjności. Mimo że BlackBerry należą do typowych przykładów smartfonów, ich użytkownicy nie zawsze mieli do dyspozycji przeglądarkę właściwą tak zaawansowanym urządzeniom. Firma Research in Motion (RIM) wprost doskonale udokumentowała funkcje przeglądarki BlackBerry Browser w kolejnych wydaniach systemu operacyjnego swoich urządzeń, zatem przynajmniej zrozumienie ich ograniczeń nie stanowi najmniejszego problemu.

Witryna BlackBerry Developer Zone (*http://na.blackberry.com/eng/developers/*) jest utrzymywana przez społeczność programistów aplikacji mobilnych dla platformy BlackBerry. Firma Research in Motion (RIM) udostępnia programistom rozwiązań mobilnych rozbudowaną dokumentację, narzędzia, emulatory i fora. Istnieje możliwość pobrania dokumentacji przeglądarki BlackBerry Browser (*http://docs.blackberry.com/en/developers/subcategories/?userType=21&category=BlackBerry+Browser*) dla kilku ostatnich wydań tej platformy (od wersji 4.2 do wydanej ostatnio wersji 5.0).

Przeglądarka BlackBerry Browser implementuje standardy internetu mobilnego już od wczesnych etapów ewolucji tej platformy. Okazuje się jednak, że do wydania platformy BlackBerry 4.6 w 2008 roku tradycyjne standardy internetowe były albo ignorowane, albo implementowane tylko w części, co rodziło frustrację wśród programistów rozwiązań mobilnych zmuszanych do odpowiedniej optymalizacji swoich stron internetowych. Co więcej, początkowo zaimplementowano tylko wybrane elementy wczesnych standardów internetu mobilnego, takie jak Wireless

CSS. W tej sytuacji firma RIM wybrała bodaj najlepsze rozwiązanie — dokładnie udokumentowała te częściowe implementacje, aby zainteresowanym programistom ułatwić odkrywanie, które funkcje pozostały nieobsługiwane.

Wraz z wydaniem urządzenia Bold 9000 w 2008 roku oraz urządzenia Storm 9500 (z ekranem dotykowym) w 2009 roku wprowadzono wersję 4.6 platformy BlackBerry z przeglądarką w pełni obsługującą języki HTML 4.01 i JavaScript oraz komponenty DOM Level 2 i technologię AJAX. Okazuje się jednak, że ostatnie wydanie tej przeglądarki z obsługą technologii AJAX nie zmieniło sposobu traktowania urządzeń BlackBerry przez wiele witryn internetowych — znaczna część tych witryn działa tak, jakby przeglądarka BlackBerry Browser nadal była zgodna tylko ze standardami WAP.

Mimo tych problemów, przeglądarka BlackBerry Browser jest w wielu aspektach dość innowacyjna. Począwszy od wersji 4.3, opisywana przeglądarka oferuje dwa wbudowane, niedostępne w innych przeglądarkach obiekty JavaScriptu zapewniające dostęp do rodzaju sieci i położenia wyznaczanego przez system GPS. Właściwość `blackberry.network` reprezentuje łańcuch określający rodzaj sieci mobilnej aktualnie używanej przez dany smartfon. Wartość tej właściwości zależy od rodzaju bieżącej sieci i może wskazywać na sieć CDMA, EDGE, iDEN, GPRS itp. Właściwość `blackberry.location` reprezentuje obiekt `blackberry.location` zawierający współrzędne geograficzne (według systemu GPS) danego urządzenia (oczywiście, jeśli to urządzenie obsługuje system GPS). Właściwość `blackberry.location.GPSSupported` reprezentuje wartość logiczną określającą, czy dane urządzenie mobilne obsługuje system GPS. Właściwości `blackberry.location.latitude` i `blackberry.location.longitude` reprezentują odpowiednio szerokość i długość geograficzną. Obiekt `blackberry.location` udostępnia też metody odświeżające położenie według systemu GPS i ustawiające sposoby uzyskiwania położenia na podstawie sygnału GPS. Więcej informacji na temat obiektów `blackberry.network` i `blackberry.location` języka JavaScript można znaleźć w dokumencie BlackBerry 4.3 Content Developer's Guide pod adresem *http://docs.blackberry.com/en/developers/deliverables/1369/BlackBerry_Browser_Version_4.3_Content_Developer_Guide.pdf*.

Smartfony BlackBerry, które projektuje się przede wszystkim z myślą o zastosowaniach biznesowych, obsługują wiele protokołów dostarczania treści oferowanych przez instalacje serwerów korporacyjnych BlackBerry w firmowych lub mobilnych centrach danych. Na rysunku 7.9 pokazano zrzut ekranu z wynikami wygenerowanymi przez mobilną wersję wyszukiwarki Google w przeglądarce BlackBerry Browser (dla urządzenia BlackBerry Storm 9530 z ekranem dotykowym i platformą mobilną w wersji 4.7.0).

Przeglądarka internetowa firmy Nokia instalowana w smartfonach z systemem Series 60

Smartfony z 3. i 5. wydaniem systemu Nokia Series 60 oferują przeglądarkę Nokia Web zaimplementowaną na bazie silnika WebKit przeniesionego na platformę Series 60 Symbian. Jak już wspomniano w podrozdziale „Silnik WebKit w mobilnych przeglądarkach internetowych", we wcześniejszej części tego rozdziału, silnik WebKit zapewnia smartfonom firmy Nokia obsługę wielu standardów internetowych, w tym języków XHTML i JavaScript oraz technologii AJAX i Flash Lite firmy Adobe.

Począwszy od 3. wydania systemu Series 60, przeglądarka firmy Nokia obsługuje standardy internetowe i tak zaawansowane funkcje jak układ strony, skalowanie czy możliwość otwierania wielu okien. Obsługę technologii AJAX po raz pierwszy zaimplementowano właśnie w 3. wydaniu,

Rysunek 7.9. Wyniki wyszukiwarki Google w emulatorze urządzenia Blackberry Storm 9530 dla platformy BlackBerry 4.7.0

w pakiecie Feature Pack 1. W 5. wydaniu systemu Nokia Series 60 wprowadzono interfejs użytkownika przystosowany do współpracy z ekranami dotykowymi, a także mechanizmy skalowania i przybliżania stron przez użytkownika, możliwość wyświetlania stron na pełnym ekranie i obsługę klawiszy skrótów.

Wokół forum firmy Nokia (*http://www.forum.nokia.com*) powstała społeczność programistów rozwiązań mobilnych i aplikacji dla telefonów Nokia. Po rejestracji na tej witrynie (rejestracja jest darmowa) programiści mogą przeglądać i pobierać artykuły techniczne, specyfikacje, emulatory i przykładowe fragmenty kodu. Emulatory platformy Nokia Series 60 obejmują wszystkie wstępnie instalowane aplikacje mobilne, w tym przeglądarkę Nokia Web Browser. Istnieje możliwość pobrania emulatorów dla każdej z wydanych wersji platformy Nokia Series 60 (a także dla innych ważnych systemów operacyjnych firmy Nokia, takich jak Series 40).

Więcej informacji na temat wszystkich technologii zaimplementowanych w przeglądarce firmy Nokia (szczególnie w przeglądarce na bazie silnika WebKit instalowanej w smartfonach z systemem Series 60) można znaleźć w sekcji Web Technologies witryny Nokia Forum (*www.forum.nokia.com/Technology_Topics/Web_Technologies/*). Przenoszenie silnika WebKit na platformę Series 60 Symbian jest realizowane w ramach projektu open-source; informacje na temat tego projektu można znaleźć na stronie *http://trac.webkit.org/wiki/S60Webkit*. Strona zawiera odsyłacze do stanu projektu, strony pobierania kodu źródłowego systemu Symbian (w języku C++) oraz materiały pokazujące, jak budować biblioteki silnika WebKit.

Na rysunku 7.10 pokazano przeglądarkę Nokia Web Browser w emulatorze smartfonu N97.

Rysunek 7.10. Przeglądarka Nokia Web Browser w emulatorze urządzenia Nokia N97 z 5. wydania platformy Series 60

Internet Explorer Mobile dla systemu Windows Mobile

Internet Explorer Mobile (IE Mobile) jest przeglądarką internetową instalowaną we wszystkich urządzeniach z systemem operacyjnym Windows Mobile. Przeglądarka Internet Explorer 6 (dostarczana wraz z systemem Windows Mobile 6) obsługuje następujące standardy internetowe.

- **Znaczniki:** HTML 4.01, XHTML 1.0 i 1.1, XHTML-MP, XHTML Basic i WML.
- **Style:** CSS 2.1, CSS Mobile Profile 1.0 i Wireless CSS 1.1.
- **Skrypty:** JScript 5.6 (zgodny ze standardem ECMAScript 3); DOM 1 oraz elementy modeli DOM 2 i DOM 3; AJAX.
- **Inne:** Google Gears (IE Mobile 4.01 i nowsze).

Przeglądarka Internet Explorer Mobile implementuje, co prawda, technologię AJAX, jednak procedura tworzenia obiektu żądania wymaga stosowania nieco innej składni niż w pozostałych przeglądarkach instalowanych w smartfonach. W przeglądarce Internet Explorer Mobile na potrzeby żądań technologii AJAX wykorzystuje się obiekt ActiveX nazwany `Microsoft.XMLHTTP`, spójny z pozostałymi przeglądarkami internetowymi firmy Microsoft, ale niezgodny z przeglądarkami innych producentów. Przeglądarka Internet Explorer Mobile tworzy obiekt żądania technologii AJAX, stosując następujące wyrażenie JavaScriptu:

```
var req = new ActiveXObject('Microsoft.XMLHTTP');
```

Większość innych przeglądarek mobilnych tworzy dla technologii AJAX obiekt żądania `XMLHttpRequest`, stosując następującą konstrukcję składniową języka JavaScript:

```
var req = new XMLHttpRequest();
```

Na listingu 5.3 w rozdziale 5. przedstawiono przykład funkcji tworzącej żądanie technologii AJAX i przystosowanej do działania w wielu przeglądarkach (korzystającej z obiektu żądania zdefiniowanego dla bieżącej przeglądarki mobilnej).

W przeglądarce Internet Explorer Mobile do określania, czy dokument internetowy zawiera znaczniki zoptymalizowane pod kątem urządzeń mobilnych z ekranami określonej szerokości, służy znacznik `MobileOptimized <meta>`. Poniższy przykład znacznika `<meta>` określa, że preferowaną szerokością ekranu jest 240 pikseli:

```
<meta name="mobileoptimized" content="240" />
```

Kiedy przeglądarka Internet Explorer Mobile rozpoznaje znacznik `MobileOptimized <meta>` w tej formie, przyjmuje, że dany dokument witryny internetowej zoptymalizowano z myślą o urządzeniach mobilnych. Przeglądarka Internet Explorer Mobile wyświetla tę stronę, ale nie skaluje jej w sposób, do którego przyzwyczaiły nas tradycyjne przeglądarki internetowe. Przeglądarka Internet Explorer Mobile wyświetli tę stronę w pojedynczej kolumnie o szerokości równej wartości atrybutu `content` (więcej informacji o znaczniku `MobileOptimized <meta>` umożliwiającym oznaczanie dokumentów jako zoptymalizowane z myślą o urządzeniach mobilnych można znaleźć w rozdziale 12.).

W ramach programu Microsoft Developer Network (MSDN) udostępniono dokumentację przeglądarki Internet Explorer Mobile i przykładowe fragmenty kodu opracowanego z myślą o tej przeglądarce (*http://msdn.microsoft.com/pl-pl/library/bb159821.aspx*). Artykuły MSDN dotyczą przeglądarki Internet Explorer Mobile 6 i nowszych wersji.

Urządzenia z systemem operacyjnym Windows Mobile coraz częściej oferowane są ze wstępnie zainstalowanymi przeglądarkami innych producentów (np. Opera Mobile), które pełnią funkcje przeglądarek domyślnych. Przeglądarka Internet Explorer Mobile także jest instalowana w tych urządzeniach, jednak jej uruchomienie zwykle jest utrudnione i wymaga użycia menu aplikacji.

Przeglądarki Opera Mini i Opera Mobile

Opera to niezależna, norweska firma tworząca oprogramowanie — bodaj najbardziej znanym produktem tej firmy jest przeglądarka Opera występująca w dwóch wersjach mobilnych: Opera Mini i Opera Mobile. Opera Mini i Opera Mobile to odmienne przeglądarki mobilne kierowane do różnych rodzajów urządzeń mobilnych.

Opera Mini jest lekką aplikacją przeglądarki napisaną w środowisku Java Platform, Micro Edition (Java ME lub J2ME) instalowaną w zwykłych telefonach. Przeglądarka Opera Mini jest częścią rozwiązania klient-serwer umożliwiającego użytkownikom popularnych urządzeń mobilnych przeglądanie dowolnych witryn mobilnych i witryn tradycyjnych, nawet jeśli witryny te korzystają z zaawansowanych funkcji uniemożliwiających ich przeglądanie za pomocą przeglądarek wbudowanych.

Klient Opera Mini jest instalowany w wielu popularnych urządzeniach mobilnych. Z perspektywy użytkownika końcowego Opera Mini przypomina tradycyjną przeglądarkę internetową, tyle że działającą w telefonie mobilnym z ograniczoną ilością zasobów. Okazuje się jednak, że zgodność tej przeglądarki z wieloma różnymi urządzeniami jest możliwa dzięki transkodowaniu

dokumentów i rezygnacji z części funkcji. Klient Opera Mini komunikuje się z serwerem Opera, który wykonuje niezbędne operacje na żądanym dokumencie i odsyła do klienta odpowiednio zoptymalizowaną wersję tego dokumentu. Serwer tłumaczy kod języka HTML na język OBML (ang. *Opera Binary Markup Language*; w praktyce jest to obraz z obszarami dostępnymi do klikania) i odsyła kod OBML-a do przeglądarki, która wyświetla tak przekształconą stronę. Przeglądarka Opera Mini obsługuje język JavaScript i technologię AJAX, ale nie dopuszcza do wykonywania tego rodzaju skryptów w urządzeniach mobilnych. Wszystkie zaawansowane funkcje przeglądarki są wykonywane po stronie serwera. Jak nietrudno odgadnąć, przeglądarka Opera Mini obsługuje model zdarzeń JavaScriptu ograniczony tylko do zdarzeń, które mogą być przechwytywane po stronie klienta i przesyłane na serwer w celu właściwego przetworzenia. Przetwarzanie w tle i inicjalizowanie zdarzeń JavaScriptu przy użyciu licznika czasowego nie jest możliwe.

Uwaga! Firma Opera Software przeniosła przeglądarkę Opera Mini na platformę Android. (Aplikacja Opera Mini jest dostępna za darmo w sklepie Android Market). W kontekście smartfonów firma Opera przekonuje, że przeglądarka Opera Mini pozwala oszczędzać czas i pieniądze, ponieważ stosowany transkoder kompresuje strony internetowe o blisko 90%. Jednak z naszego doświadczenia wynika, że korzystanie z tego mechanizmu transkodującego w systemie Android nie ma większego sensu. Ta zaawansowana platforma mobilna zawiera rozbudowaną przeglądarkę zaimplementowaną przy użyciu silnika WebKit, zatem większość użytkowników decyduje się na stosowanie właśnie tej przeglądarki (chyba że koszty przesyłania danych lub przepustowość stanowią poważny problem).

Na rysunku 2.19 w rozdziale 2. pokazano zrzut ekranu emulatora przeglądarki Opera Mini dostępnego na stronie internetowej *www.opera.com/mini/demo/* (korzystanie z tego emulatora na komputerze biurkowym wymaga zainstalowanej platformy Java). Dokumentację obsługi języka JavaScript w przeglądarce Opera Mini 4 można znaleźć pod adresem *http://dev.opera.com/articles/view/javascript-support-in-opera-mini-4/*.

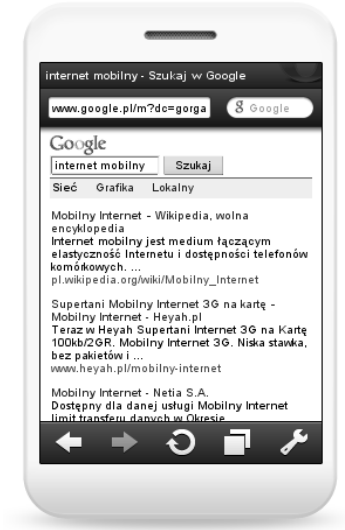
Przeglądarka Opera Mobile jest programem zupełnie innym od przeglądarki Opera Mini. Opera Mobile to autonomiczna przeglądarka internetowa zgodna ze standardami i opracowana z myślą o urządzeniach mobilnych z systemami Windows Mobile i Nokia Series 60.

Obsługa takich rozwiązań jak przeglądanie wielu zakładek, przybliżanie dokumentów czy optymalizacja pod kątem ekranów mobilnych przyspiesza i ułatwia użytkownikom pracę z tą przeglądarką. Opera Mobile obsługuje następujące standardy internetowe.

- **Znaczniki:** XHTML 1.0; HTML 4.01 i elementy standardu HTML 5; WML 1.3 i 2.0.
- **Style:** CSS 2 i 3.
- **Skrypty:** JavaScript, DOM Level 2 i AJAX.
- **Inne:** Google Gears (Opera Mobile 9.5 i nowsze).

Firma Opera nie udostępnia emulatora przeglądarki Opera Mobile. Jeśli jednak masz dostęp do systemu Windows Mobile lub urządzenia mobilnego firmy Nokia, możesz uzyskać wygląd stron tradycyjnych witryn internetowych zbliżony do formy, w jakiej prezentowane są w przeglądarce Opera Mobile, instalując biurkową przeglądarkę Opera (*www.opera.com/download/*). Wystarczy otworzyć jakąś stronę tradycyjnej witryny internetowej, odpowiednio zmniejszyć rozmiar okna i wybrać z menu *Widok* opcję *Dopasuj do szerokości*. Przeglądarka Opera wyświetli wówczas wybraną stronę w sposób zbliżony do przeglądarki Opera Mobile, ponieważ obie przeglądarki korzystają z tego samego silnika wyświetlania stron. (W przeglądarce mobilnej, oczywiście, mamy do dyspozycji nieporównanie mniejszą liczbę czcionek).

Na rysunku 7.11 pokazano wyniki wyświetlone przez mobilną wersję wyszukiwarki Google w emulatorze przeglądarki Opera Mini 5.



Rysunek 7.11. Wyniki mobilnej wyszukiwarki Google w emulatorze przeglądarki Opera Mini 5

Na rysunku 7.12 pokazano przykład tradycyjnej witryny internetowej (*http://www.helion.pl*) otwartej w zmniejszonym oknie przeglądarki Opera (w formie zbliżonej do tej, która zostałaby wyświetlona przez przeglądarkę mobilną).

Rysunek 7.12. Widok małego ekranu przeglądarki Opera przypominający przeglądarkę mobilną Opera Mobile

ĆWICZENIE 7.: PRZEGLĄD PRZEGLĄDAREK INSTALOWANYCH W SMARTFONACH

Użyj technik opisanych w podrozdziałach „Popularne techniki tworzenia witryn dla przeglądarek instalowanych w smartfonach" i „Silnik WebKit w mobilnych przeglądarkach internetowych" do wzbogacenia stron swojej istniejącej już witryny internetowej o standardy internetowe, skrypty języka JavaScript i rozszerzenia standardu CSS obsługiwane przez silnik WebKit. Zacznij od strony internetowej zgodnej ze standardami mobilnymi — udoskonal kod języka znaczników, skrypty i style, korzystając z większych możliwości przeglądarek dla smartfonów (przeglądarek opracowanych na bazie silnika WebKit). Dodawaj nowe funkcje aż do momentu, w którym funkcje i użyteczność zmienionego dokumentu będą dostatecznie zoptymalizowane. Spróbuj wykorzystać choć część rozszerzeń CSS silnika WebKit. W trakcie realizacji tego ćwiczenia spróbuj wykonać następujące zadania.

- Otwórz zmienioną stronę witryny mobilnej w przeglądarkach korzystających z silnika WebKit w urządzeniu iPhone oraz urządzeniach z systemami Android, webOS i Nokia Series 60. Jeśli nie dysponujesz tymi urządzeniami, użyj odpowiednich emulatorów.
- Otwórz zmienioną stronę witryny mobilnej w przeglądarkach smartfonów, których nie zaimplementowano na bazie silnika WebKit, np. w przeglądarce urządzenia BlackBerry oraz przeglądarkach Internet Explorer Mobile i Opera Mobile.
- Otwórz zmienioną stronę witryny mobilnej w takich tradycyjnych przeglądarkach korzystających z silnika WebKit jak Google Chrome czy Apple Safari.

Odpowiedz na następujące pytania.

1. Na ile spójne są układ i doznania użytkownika Twojej mobilnej witryny internetowej otwieranej w wielu różnych przeglądarkach korzystających z silnika WebKit? Czy widać różnice pomiędzy tymi przeglądarkami a przeglądarkami, które nie używają silnika WebKit? Czy występują jakieś różnice dzielące wygląd tej witryny otwieranej w przeglądarkach mobilnych i tradycyjnych?
2. W jakim stopniu wykorzystanie dodatkowych standardów internetowych i rozszerzeń obsługiwanych przez silnik WebKit wpłynęło na użyteczność, wydajność i rozmiar dokumentów Twojej mobilnej witryny internetowej?
3. Jak strony zoptymalizowane pod kątem smartfonów prezentują się w przeglądarkach mobilnych instalowanych w bardziej popularnych, zwykłych telefonach? Jakie zmiany można by wprowadzić, aby zabezpieczyć doznania użytkowników tego rodzaju przeglądarek?

Podsumowanie

W tym rozdziale wprowadzono zaawansowane funkcje przeglądarek instalowanych w smartfonach, które czynią z tych urządzeń bardzo atrakcyjne platformy dla bogatych mobilnych aplikacji internetowych. Przeglądarki instalowane w smartfonach obsługują typowe elementy języków znaczników i zdarzenia JavaScriptu, które znacznie rozszerzają możliwości tworzenia interaktywnej treści dokumentów mobilnych. Zaawansowany silnik przeglądarek WebKit obsługuje standardy internetowe i jako taki stanowi podstawę dla wielu domyślnych przeglądarek mobilnych instalowanych w smartfonach. Wspomniany silnik umożliwia programistom opracowanie jednej zoptymalizowanej wersji dla smartfonów iPhone oraz urządzeń mobilnych z systemami Android, webOS i Nokia Series 60.

Przeglądarki mobilne instalowane w smartfonach oferują zróżnicowane zbiory funkcji i odmienne zbiory implementowanych standardów. W tym rozdziale omówiono funkcje przeglądarek internetowych oferowane wraz z telefonach iPhone oraz urządzeniami z systemami Android, webOS, BlackBerry, Nokia Series 60 i Windows Mobile. Wyjaśniono różnice dzielące dwie popularne przeglądarki mobilne firmy Opera Software: Operę Mini (instalowaną w popularnych, prostych urządzeniach mobilnych i korzystającą z mechanizmu transkodowania dokumentów) oraz Operę Mobile (zgodną z licznymi standardami internetowymi przeglądarkę dla smartfonów).

W następnym rozdziale omówimy techniki optymalizacji przetwarzania końcowego (ang. *post-processing*) języka znaczników, arkuszy stylów, skryptów i obrazów w ramach dokumentów mobilnych. Taka optymalizacja ma na celu minimalizowanie ilości przesyłanych danych, podnoszenie wydajności i ułatwianie buforowania danych po stronie przeglądarek mobilnych.