

Rozwijaj i sprzedawaj aplikacje dla Windows 8!

JavaScript

Aplikacje dla **Windows 8**



Chris **Sells**, Brandon **Satrom**,
Don **Box**

Tytuł oryginału: Building Windows 8 Apps with JavaScript

Tłumaczenie: Jakub Hubisz

ISBN: 978-83-246-7564-7

Authorized translation from the English language edition, entitled: BUILDING WINDOWS 8 APPS WITH JAVASCRIPT; ISBN 0321861280; by Chris Sells, and Brandon Satrom, and Don Box; published by Pearson Education, Inc, publishing as Addison Wesley.

Copyright © 2013 Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

Polish language edition published by HELION S.A. Copyright © 2014.

The .NET logo is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries and is used under license from Microsoft. Microsoft, Windows, Visual Basic, Visual C#, and Visual C++ are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries/regions.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION

ul. Kościuszki 1c, 44-100 GLIWICE

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<ftp://ftp.helion.pl/przyklady/jascw8.zip>

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/jascw8>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)



Spis treści

Słowo wstępne Chris Anderson 13

Słowo wstępne Rey Bango 15

Wstęp 17

Podziękowania 21

O autorach 25

1 Witaj, Windows 8! 27

Twoja pierwsza aplikacja ze Sklepu Windows 28

Pierwsze kroki z Visual Studio 2012 31

Kontrolki, bindowanie i stylowanie w Blend 38

Nawigacja 43

Obsługa sieci w WinJS i WinRT 48

Szablon Aplikacja podziału 51

Reszta 54

Gdzie jesteśmy? 55

2 Bindowanie i kontrolki 57

Bindowanie 57

Bindowanie obiektów 58

Inicjalizatory 64

Listy bindowania 65

Sortowanie i filtrowanie 67

Grupowanie 69

Szablony 70

Kontrolki 73

Elementy HTML 73

Kontrolki WinRT 74

Kontrolki WinJS 75

Kontrolki niestandardowe 78

Gdzie jesteśmy? 83

3 Układ 85

- Układy: poskromienie macierzy urządzenia 85
 - Windows 8: wybór konsumenta bez tyranii urządzeń* 86
 - Układ w Windows 8* 86
 - Praca z rozmiarami ekranów* 88
 - Orientacja* 95
 - Stany widoku* 97
- Wykorzystywanie możliwości układania elementów w CSS3 99
 - Specyfikacja układu siatkowego CSS* 100
 - Układy adaptacyjne dla zawartości aplikacji* 103
- Tworzenie interfejsów adaptacyjnych z CSS i WinJS 103
 - Wykorzystanie Flexboks w interfejsach adaptacyjnych* 103
 - Zastosowanie układu wielokolumnowego w interfejsach adaptacyjnych* 105
 - Tworzenie adaptacyjnych kolekcji za pomocą ListView* 108
 - Reagowanie w JavaScriptcie na zmiany układu* 110
- Gdzie jesteśmy? 113

4 Typografia 115

- Typografia w aplikacjach ze Sklepu Windows 115
 - Segoe UI* 116
 - Cambria* 117
 - Calibri* 117
 - Czcionki web w CSS3* 119
 - Wykorzystanie CSS do dostosowania typografii* 122
- Praca z ikonografią 126
- Wykorzystanie czcionek ikon w aplikacji ze Sklepu Windows i manipulacja tymi czcionkami 133
- Gdzie jesteśmy? 138

5 Media 139

- Praca z audio i wideo 139
- Pierwsze kroki z mediami w Windows 8 140
- Kontrolowanie wyglądu mediów i tworzenie niestandardowych kontroltek 142
- Dodawanie napisów do filmu 145
- Dodawanie efektów wizualnych 148
- Praca z dźwiękiem w aplikacjach ze Sklepu Windows 151
 - Tworzenie dźwięku w tle* 151
- Praca z bibliotekami mediów użytkownika przy użyciu kontroltek wybierania plików 154
 - Wybór wielu plików* 158
- Inne typy wybierania plików 160
- Praca z uchwyconymi mediami 162
- Dodanie obsługi dla Play To 165
- Gdzie jesteśmy? 168

6 Rysowanie i animacja 169

- Grafika w HTML5 z SVG i Canvas 169
 - Wprowadzenie do SVG* 169
 - Wprowadzenie do Canvas* 173
 - Wybór pomiędzy Canvas i SVG* 177

Manipulacja pikselami	178
<i>Manipulacja pikselami za pomocą Canvas</i>	178
<i>Manipulacja pikselami za pomocą Windows.Graphics.Imaging</i>	180
Animacje w aplikacjach ze Sklepu Windows	183
<i>Szybka i płynna animacja w aplikacjach ze Sklepu Windows</i>	183
<i>Transformacja i animacja przy użyciu CSS</i>	184
<i>Praca z biblioteką WinJS Animation</i>	187
Gdzie jesteśmy?	190

7 Stan aplikacji 191

Ustawienia	191
<i>Panel ustawień</i>	193
Cykl życia	201
<i>Metody pomocnicze dla zdarzeń cyklu życia WinJS</i>	203
<i>Sesje</i>	203
<i>Debugowanie sesji</i>	206
<i>Funkcje pomocnicze WinJS dla sesji</i>	209
Pliki	211
<i>Funkcje pomocnicze WinJS dla plików</i>	213
Biblioteki	214
<i>Aktywacja plików</i>	216
<i>Okna wybierania plików</i>	217
Gdzie jesteśmy?	221

8 Operacje sieciowe 223

Możliwości sieciowe	223
Sieć mobilna	224
Obiekt XMLHttpRequest	227
<i>Analiza wyników XML</i>	228
<i>Postęp i błędy pobierania</i>	228
<i>Analiza wyników JSON</i>	229
Syndykacja	231
Transfer danych w tle	232
Treści WWW	236
<i>Treść HTML</i>	237
<i>Hosting elementów iframe</i>	237
<i>Kontekst WWW</i>	239
Gdzie jesteśmy?	242

9 Kontrakty powłoki 243

Powłoka Windows 8	243
Kontrakty	244
Kontrakt wyszukiwania	246
<i>Implementacja wyszukiwania</i>	246
<i>Sugestie wyszukiwania</i>	251
<i>Kontrakty udostępniania</i>	252
<i>Udział docelowy</i>	255
<i>Wgląd w udostępnione dane</i>	261
<i>Raportowanie postępu udostępniania</i>	265

Kontrakt kontaktów	265
Selektor kontaktów	265
Dostawcy kontaktów	267
Debugowanie dostawcy kontaktów	272
Gdzie jesteśmy?	273

10 Integracja z powłoką 275

Dynamiczne kafelki	275
Kafelek Twojej aplikacji	276
Aktualizacje kafelka	276
Aktualizacje małych i dużych kafelków	279
Obrazy kafelka	280
Odwracanie kafelka	281
Zaplanowane aktualizacje kafelka	282
Kafelki dodatkowe	283
Znaczki	286
Zadania w tle	288
Wyzwalanie zadania w tle	288
Tworzenie zadania w tle	290
Aplikacje ekranu blokowania	291
Zapobieganie duplikacji zadań	293
Powiadomienia w pasku przewijania	294
Aktywacja aplikacji za pomocą powiadomienia	296
Zaplanowane powiadomienia	297
Gdzie jesteśmy?	297

11 Interakcja z urządzeniem 299

Wprowadzenie do dotyku	299
Przyjazne w dotyku kontrolki HTML	300
Przyjazne w dotyku kontrolki WinJS	302
Tworzenie przyjaznych w dotyku aplikacji z brzegami ekranu	303
Tworzenie przyjaznych w dotyku interakcji za pomocą kontrolki SemanticZoom	305
Interakcje za pomocą myszy i klawiatury	310
Wykorzystanie możliwości urządzenia	310
Deklarowanie możliwości urządzenia	311
Praca z urządzeniami rejestrującymi	311
Dodawanie możliwości drukowania	313
Praca z danymi lokalizacyjnymi	316
Użycie obiektu Geolocator	316
Obserwacja zmian położenia	318
Użycie danych lokalizacyjnych z mapami Bing Maps	319
Symulowanie informacji o lokalizacji	320
Praca z czujnikami	320
Praca z czujnikiem oświetlenia	323
Praca z przyspieszeniem	324
Praca z kompasem	325

Praca z prostym czujnikiem położenia	327
Praca z innymi czujnikami	328
Gdzie jesteśmy?	328

12 Natywne rozszerzenia kodu 329

Wiele języków, jedna aplikacja	329
Pierwsze kroki	331
WinRT i środowisko JavaScript	334
Klasy WinRT	335
<i>Klasy i metody</i>	336
<i>Metody i wyjątki</i>	338
<i>Klasy i właściwości</i>	340
Obiekty WinRT	341
<i>Obiekty i uchwyt</i>	342
Typy WinRT w środowiskach C++/CX i JavaScript	343
<i>Ciągi znaków</i>	347
<i>Tabele</i>	349
<i>Typy wartości WinRT</i>	350
Delegaty i funkcje	351
<i>Funkcje lambda C++11</i>	352
<i>Tworzenie delegatów WinRT z funkcji lambda C++11</i>	354
Zdarzenia	355
Współbieżność i asynchroniczność	357
Gdzie jesteśmy?	363

13 Zarabianie pieniędzy 365

Przygotowanie aplikacji do przesłania	365
<i>Utworzenie konta programisty</i>	366
<i>Rezerwacja nazwy aplikacji</i>	366
<i>Przygotowanie aplikacji do lokalnych testów</i>	368
<i>Uruchomienie zestawu Windows App Certification Kit (WACK)</i>	370
Przesyłanie aplikacji do Sklepu Windows	373
<i>Zakończenie procesu przesyłania aplikacji do Sklepu Windows</i>	373
<i>Oczekiwanie na certyfikację</i>	380
<i>Obsługa odrzucenia aplikacji</i>	381
<i>Przesyłanie aktualizacji</i>	382
Umieszczanie reklam	382
<i>Zasady dotyczące umieszczania reklam w aplikacjach Windows 8</i>	383
<i>Korzystanie z pakietu Windows 8 Ads SDK</i>	384
<i>Zastosowanie mediów w reklamach</i>	384
<i>Umieszczanie reklam tekstowych</i>	387
Implementacja okresu próbnego aplikacji	389
<i>Wprowadzenie do Sklepu Windows i symulatora</i>	389
<i>Symulowanie i testowanie okresu próbnego</i>	391
Oferty w aplikacji	393
<i>Implementacja funkcjonalności oferty w aplikacji</i>	393
<i>Definiowanie ofert w procesie przesyłania aplikacji do Sklepu Windows</i>	397
Projektowanie komercyjnych aplikacji	398

Sprzedaż aplikacji i zarządzanie nią	398
Śledzenie aplikacji na pulpicie Sklepu Windows	398
Eksponowanie aplikacji w Sklepie Windows	399
Odbiór zapłaty	400
Gdzie jesteśmy?	401

A JavaScript dla programistów pracujących w językach z rodziny C 403

Witaj, świecie	403
Separacja potrzeb	404
Wykorzystanie identyfikatora id jako obiektu	405
Aktywacja WinJS	406
Wartości i typy	407
Operatory	408
Obiekty	409
Daty	410
Wyrażenia regularne	410
Tablice	411
Prototypy obiektów (klasy)	412
Konstruktory	413
Prototypy	413
Dziedziczenie prototypów	415
Właściwości i metody statyczne	416
Definiowanie klas za pomocą WinJS	416
Funkcje	417
Argumenty funkcji	418
Call i Bind	419
Domknięcia	420
Dane wyjściowe debugowania	421
Ustalanie zasięgu	421
Wynoszenie	422
Moduły	422
Przestrzenie nazw	423
Przestrzenie nazw WinJS	423
Tryb standardów — strict	423
Serializacja	425

B Rzut oka na style i prezentację 427

Wykorzystanie kodu HTML do tworzenia zawartości i struktury aplikacji	427
Czym jest HTML?	428
Nowości w HTML5	429
Wykorzystanie CSS do zdefiniowania wyglądu strony	432
Czym jest CSS?	433
Gdzie należy definiować style CSS	441
Kaskadowość reguł CSS	442
CSS w aplikacjach ze Sklepu Windows	444
Nadpisywanie domyślnych stylów aplikacji ze Sklepu Windows	446

Skorowidz 449

Interakcja z urządzeniem

SYSTEM WINDOWS 8 JEST POŁĄCZENIEM STARYCH I NOWYCH IDEI; jest to współistnienie interakcji z komputerem, do których jesteś przyzwyczajony, z nowym punktem widzenia wprowadzonym wraz z epoką smartfonów i tabletów. Mysz, klawiatura i pióro cyfrowe są wciąż pierwszoplanowymi aktorami w interakcji i obsłudze komputera, lecz w Windows 8 do tych tradycyjnych metod dołączył dotyk. A dotyk to nie tylko udawanie myszy palcem z podążającym za nim kursorem. Mam tu na myśli metaforę prawdziwego dotyku i interakcji, czyli przeciąganie, przerzucanie, obracanie i inne czynności. Twój komputer wyposażony w system Windows 8 staje się czymś, czym możesz manipulować, jak nigdy dotąd. Cały system operacyjny powstał z myślą o dotyku.

Twój komputer z systemem Windows 8 jest wyposażony, oprócz ekranu dotykowego, w szeroki wachlarz urządzeń i czujników, które mogą Ci powiedzieć więcej, niż to się wydaje możliwe, o Twojej lokalizacji, komputerach i treści wokół Ciebie, jasności otoczenia, a nawet o tym, czy trzymasz (lub przynosisz) swój komputer. Co więcej, cała ta informacja jest dla Ciebie, jako programisty, na wyciągnięcie ręki.

W tym rozdziale zajmę się szczegółowo interakcją z urządzeniem. Zacznę od tego, jak utworzyć dotykową aplikację, korzystając z możliwości standardowo oferowanych przez Windows 8, a także różnych narzędzi i kontrolki sprawiających, że aplikacja będzie przyjazna w dotyku, a w razie potrzeby umożliwiała interakcję za pomocą myszy, klawiatury lub pióra. Następnie omówię urządzenia i czujniki oraz sposób, w jaki Windows 8 umożliwia programistom dostęp do tych narzędzi i tworzenie aplikacji komunikujących się ze światem wokół użytkownika.

Wprowadzenie do dotyku

Kiedy w 1963 roku zostały wymyślone przez Douglasa Engelbarta mysz i graficzny interfejs użytkownika, udoskonalone przez firmę Xerox PARC w roku 1970, wydawało się, że jest to rewolucja w epoce dominacji terminalu i aplikacji tekstowych obsługiwanych jedynie przez klawiaturę. Przez ostatnie 40 lat komputerem stacjonarnym i laptopem rządziła mysz, a nasze oczekiwania wobec użytkowanego oprogramowania (lub też wrażenia użytkownika, jeżeli wolisz) były podporządkowane temu małemu niesfornemu stworzeniu leżącemu obok klawiatury. Potęga myszy wciąż pozostaje niezachwiana i, pomimo kilku dekad dyskusji o naturalnym interfejsie użytkownika i alternatywnych sposobach obsługi komputera, wydaje się, że miejsce myszy obok komputerów i laptopów będzie zapewnione przynajmniej jeszcze przez jakiś czas.

Lecz dzisiaj informatyka to nie tylko komputery stacjonarne i laptopy. W coraz większym stopniu to, co określamy mianem „informatyki”, dzieje się poza biurkiem, w miejscach, gdzie nawet laptop nie ma

zastosowania. Ten nowy rodzaj informatyki — **mobilna informatyka** — może pojawiać się wszędzie, w pociągu, w parku lub na kempingu, gdy czekasz całą noc na bilet na koncert Justina Biebera. Mobilna informatyka, jak sama nazwa wskazuje, jest informatyką w ruchu.

Nie znaczy to, że komputery stacjonarne, laptopy i myszy odchodzą w przeszłość, ale raczej to, że coraz więcej czasu z informatyką spędza się, wykonując niestandardowe czynności, korzystając ze smartfonów i tabletów specjalnie zaprojektowanych do użytku przenośnego. Jeżeli wierzyć statystykom, prawdopodobnie posiadasz smartfon, czy to będzie iPhone, Windows Phone, Android lub Blackberry. Zapewne masz również tablet iPad lub Android. Wiesz już, że te urządzenia w głównej mierze są obsługiwane palcami. Za podstawę tej alternatywnej informatyki został wybrany **dotyk** i w dużej mierze jest on spełnieniem wielu obietnic, jakie zostały dane w dyskusjach o naturalnym interfejsie użytkownika toczonech w czasie ubiegłych lat.

Dlaczego dotyk? Chyba dlatego, że jest to naturalny odruch. To właśnie poprzez dotyk komunikujemy się codziennie z otaczającym nas fizycznym światem. Jest to przyjazny, szybko przyswajany niemal przez każdego sposób interakcji. Aby się przekonać, co mam na myśli, podaj tablet trzyletniemu dziecku i obserwuj, co się będzie działo. U moich dzieci reakcja była natychmiastowa, a po kilku próbach i błędach odkryły, jak należy właściwie posługiwać się urządzeniem. Jeżeli ten sam eksperyment powtórzysz z myszą i klawiaturą, zauważysz ogromny kontrast w naturalnym komforcie i przyjazności obsługi¹.

Jak już wspomniałem, popularność dotyku nie oznacza odejścia myszy i klawiatury do lamusa. Dla większości użytkowników obie formy komunikacji są bardzo ważne i mają swoje zalety oraz wady. Uważam, że urządzenia dotykowe są idealne do zastosowań konsumpcyjnych, takich jak gry komputerowe. Nie są jednak efektywne w takich pracach jak pisanie tekstu, programowanie lub tworzenie projektów w programie Photoshop. Jestem jednym z rosnącej liczby użytkowników, którzy potrzebują oprócz urządzeń dotykowych do konkretnych zastosowań również takich, jakie realizują inne czynności, w zależności od tego, czy łatwiej je wykonać przez dotyk, czy za pomocą myszy.

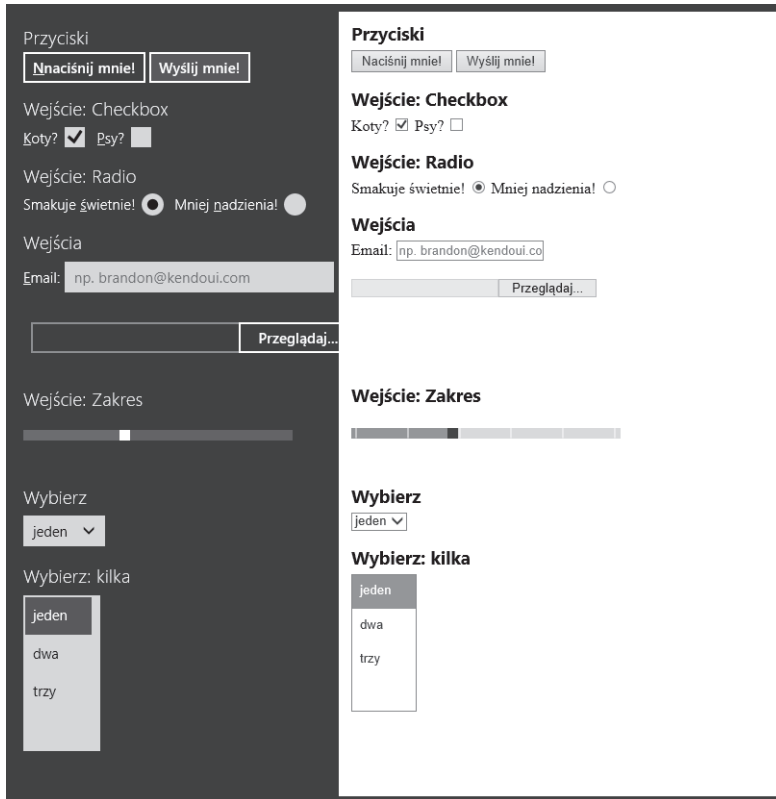
System Windows 8 został zaprojektowany z tą właśnie myślą. Jest pierwszym systemem operacyjnym przystosowanym do dotyku jako podstawowego sposobu obsługi, ale nieodrzucającym interakcji, których wciąż potrzebują użytkownicy w szczególnych przypadkach. Idea podporządkowania dotykowi z zachowaniem interakcji za pomocą myszy i klawiatury jest zaszyta w strukturze DNA systemu Windows 8, a od Ciebie jako programisty oczekuje się tworzenia aplikacji przystosowanych do dotyku z tą samą intencją, jaka przyświecała firmie Microsoft przy tworzeniu samej platformy². Na szczęście, sama platforma oferuje wielką pomoc.

Przyjazne w dotyku kontrolki HTML

Aby zapoznać się dostępnymi od ręki możliwościami dotykowej obsługi aplikacji w systemie Windows 8, skorzystajmy z przykładów z rozdziału 2., „Bindowanie i kontrolki”, i spójrzmy na domyślny wygląd powszechnie używanych kontrolki stosowanych zarówno w aplikacjach w stylu Windows 8, jak i w przeglądarce Internet Explorer 10. Zamierzam wykorzystać niektóre kontrolki w nowej aplikacji w stylu Windows 8, w której utworzymy również osobną stronę z tymi samymi znacznikami, po czym załadujemy ją do elementu `iframe`. Efekt przedstawiony jest na rysunku 11.1.

¹ Nie oznacza to, że mój syn nie potrafi posługiwać się myszą, ale bezsprzecznie łatwiej mu coś zrobić za pomocą ekranu dotykowego niż myszy.

² Microsoft przygotował znakomity wykaz zaleceń dotyczących projektowania aplikacji dotykowych, dostępny pod adresem <http://msdn.microsoft.com/en-us/library/windows/apps/hh465415.aspx> (<http://tinysells.com/286>).



Rysunek 11.1. Przyjazne w użyciu kontrolki HTML

Oba panele zawierają dokładnie ten sam kod (nie dodałem żadnych lokalnych stylów ani dostosowań stylu Windows 8) i szybko powinniśmy się przekonać, że w aplikacji w stylu Windows 8 kontrolki są większe. Niemal każda kontrolka w tym przykładzie — począwszy od przycisków wyboru, na przycisku opcji i polach tekstowych skończywszy — ma styl określony przez pakiet SDK Windows 8 i jest większa. Przyjrzyj się przyciskowi wyboru, który w większości przeglądarek ma wymiary 13×13 pikseli. Dobrze nadaje się do myszy i pióra cyfrowego, które oferują pikselową precyzję. Jeśli jednak weźmiesz pod uwagę, że ludzki palec ma przeciętnie szerokość 11 milimetrów (ok. 55 pikseli), to taka kontrolka staje się małym celem³. Jeżeli próbowałeś już posługiwać się palcem na stronie niedostosowanej do urządzeń mobilnych, na pewno odczułeś frustrację po kilku nieudanych próbach trafienia w mały cel na ekranie dotykowym.

Kontrolki w aplikacjach w stylu Windows 8 są domyślnie większe, ponieważ zostały lepiej przystosowane do dotyku. Windows 8 powiększa np. szerokość i wysokość przycisku wyboru o 8 pikseli, co pokazuję w poniższym przykładzie wziętym ze stylu WinJS (wiersz 369.):

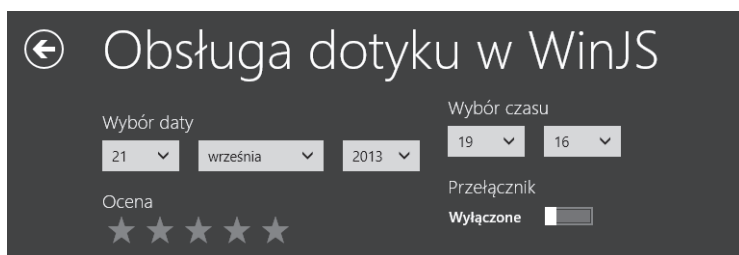
```
// ui-dark.cs
input[type=checkbox] {
  width: 21px;
  height: 21px;
  margin-right: 5px;
}
```

³ Patrz http://msdn.microsoft.com/en-US/library/windows/apps/hh465415.aspx#touch_targets (<http://tinysells.com/207>).

Ta mała zmiana stanowi dużą różnicę dla użytkowników, którzy zamiast używać swoich palców jako precyzyjnych urządzeń wskazujących, odczuwają, jak platforma, a więc i Twoja aplikacja, są zaprojektowane z myślą o ich dłoniach. Możesz korzystać z przyjaznych w dotyku kontrolerek za darmo, dołączając do swojej aplikacji jeden z domyślnych szablonów stylów (`ui-dark` lub `ui-light`). Zachęcam do używania ich i pozostania przy wielkościach kontrolerek określonych przez platformę, chyba że masz naprawdę ważny powód, aby je zmienić⁴.

Przyjazne w dotyku kontrolki WinJS

Oczywiście, domyślne kontrolki HTML nie są jedynymi przyjaznymi elementami dostępnymi w systemie Windows 8. Wszystkie kontrolki WinJS i WinRT zostały również zaprojektowane z myślą o dotyku. Jako przykład na rysunku 11.2 przedstawiam kilka kontrolerek; są to selektor daty (`DatePicker`), selektor czasu (`TimePicker`), ocena (`Rating`) oraz przełącznik (`ToggleSwitch`).



Rysunek 11.2. Przyjazne w dotyku kontrolki WinJS

Kontrolki te, podobnie jak domyślne kontrolki HTML, zostały zaprojektowane jako większe cele dla dotyku, dzięki czemu zapewniają użytkownikom znaczną wygodę. Popatrzmy na domyślny styl gwiazdek w kontrolce `Rating` WinJS (którym przypisywana jest klasa `.win-star`, gdy WinJS przetwarza wszystkie Twoje deklaracje `data-win-control`) — wiersz 1151.:

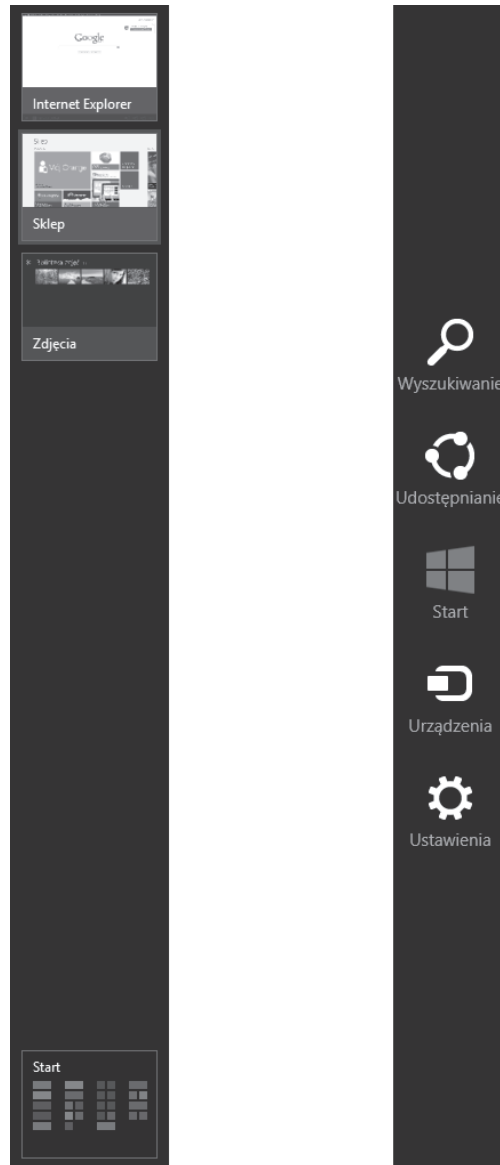
```
// ui-dark.css
.win-rating .win-star {
    -ms-flex: 1 1 auto;
    height: 28px;
    width: 28px;
    padding: 0 6px;
    font-family: "Segoe UI Symbol";
    font-size: 28px;
    ...
}
```

Kontrolka `Rating` wykorzystuje czcionkę *Segoe UI Symbol*, opisaną w rozdziale 4., „Typografia”, i ustawia wielkość pola i czcionki na 28 pikseli. W rezultacie powstaje kontrolka o wielkości odpowiedniej do użycia. Kontrolki WinJS, podobnie jak kontrolki HTML, są dopasowane do wielkości palca i gotowe do dołączenia w Twoich aplikacjach.

⁴ Jeżeli masz zamiar odstąpić od domyślnych wielkości kontrolerek, sięgnij po pomoc do instrukcji firmy Microsoft dotyczącej interakcji dotykowej, zamieszczonej pod adresem <http://msdn.microsoft.com/en-US/library/windows/apps/hh465415.aspx> (<http://tinysealls.com/209>).

Tworzenie przyjaznych w dotyku aplikacji z brzegami ekranu

Aplikacje w stylu Windows 8 są innowacyjne nie tylko dlatego, że zajmują cały ekran urządzenia, na którym są uruchamiane, ale również dlatego, że odpowiadają na interakcje pochodzące spoza niego. Jeśli korzystasz z systemu Windows 8 na ekranie dotykowym, z pewnością znasz efekt przesunięcia palcem od brzegu do środka ekranu. Przesunięcie od lewej strony umożliwia przełączanie pomiędzy uruchomionymi aplikacjami, natomiast przesunięcie od prawej strony otwiera boczny pasek menu systemu Windows 8. Na rysunku 11.3 przedstawiam wynik tych dwóch operacji.



Rysunek 11.3. Lista zadań i boczny pasek menu w systemie Windows 8

Oprócz przesuwania od lewej lub prawej strony, użytkownik może przesunąć palcem z góry na dół, aby zamknąć uruchomioną aplikację, lub z dołu do góry, by otworzyć pasek charakterystyczny dla danej aplikacji. Na rysunku 11.4 możesz zobaczyć pasek przeglądarki Internet Explorer, która wykorzystuje zarówno górną, jak i dolną część ekranu do wyświetlania kart, paska adresu i dodatkowych opcji.



Rysunek 11.4. Pasek AppBar przeglądarki Internet Explorer w systemie Windows 8

Podczas tworzenia aplikacji w stylu Windows 8 powinieneś uwzględnić brzegi górny i dolny ekranu jako elementy interfejsu użytkownika oraz jako miejsca dla określonych opcji i funkcjonalności. Najpopularniejszą funkcjonalnością, którą bez wątpienia powinieneś wykorzystać, jest pasek aplikacji AppBar, który łatwo się tworzy za pomocą kilku znaczników:

```
<div id="appBar" data-win-control="WinJS.UI.AppBar"
  data-win-options="{transient:true,autoHide:0,sticky:false}">
  <button data-win-control="WinJS.UI.AppBarCommand" id="barReset"
    data-win-options="{label:'Reset', icon:'refresh',
      section:'selection', tooltip:'Reset timera'}" >
  </button>
  <button data-win-control="WinJS.UI.AppBarCommand" id="barCancel"
    data-win-options="{label:'Anuluj', icon:'clear',
      section:'selection', tooltip:'Anuluj timer'}">
  </button>
```

```
<button data-win-control="WinJS.UI.AppBarCommand" id="barDuration"
  data-win-options="{label:'Czas trwania', icon:'clock',
  type:'flyout', flyout:'changeDuration',
  tooltip:'Zmień czas trwania'}">
</button>
</div>
```

Pasek AppBar jest po prostu znacznikiem `<div>` z atrybutem `data-win-control` o wartości `"WinJS.UI.AppBar"` oraz jednym lub kilkoma przyciskami z atrybutami `data-win-control` o wartości `"WinJS.UI.AppBarCommand"`. System Windows 8 automatycznie obsługuje wygląd, rozmieszczenie oraz animację naciśnięcia i zwolnienia tej kontrolki. Jeżeli uruchomisz aplikację zawierającą powyższe znaczniki, a następnie przesuniesz palcem z góry na dół po ekranie (lub klikniesz prawym przyciskiem myszy albo naciśniesz *Win+Z*), zobaczysz pasek AppBar w akcji (patrz rysunek 11.5).



Rysunek 11.5. Polecenia na pasku AppBar

Zauważ również, że oprócz dużych, przyjaznych w dotyku przycisków na pasku aplikacji, kontrolki są zgrupowane przy lewym i prawym brzegu ekranu. Domyślnie polecenia na pasku aplikacji są wyświetlane w kolejności od prawego brzegu ekranu, ale dopuszcza się przesunięcie niektórych poleceń na lewą stronę, kiedy tworzą osobną grupę lub jest ich dużo⁵. W tym przykładzie przesunąłem przyciski *Reset* i *Anuluj* na lewą stronę, umieszczając właściwość `section='selection'` w opcjach obu poleceń `AppBarCommand`.

W tym miejscu możesz zastanawiać się, dlaczego opcje te są umieszczone na brzegach ekranu zamiast na jego środku, gdzie wyglądałyby ładnie i symetrycznie. Odpowiedzią na to pytanie, co nie dziwi, jest dotyk. Microsoft w trakcie swoich badań odkrył, że przez większość czasu trzymamy urządzenie mobilne za jego brzegi⁶. Ten sposób jest wygodny dla większości z nas, ale daje ograniczone możliwości obsługi urządzenia. Użytkownik lub urządzenie musiałyby być w takiej pozycji, aby jedna ręka była wolna. Wykorzystanie brzegów ekranu w aplikacjach jest próbą umożliwienia obsługi aplikacji za pomocą tylko kciuka lub innego palca, dlatego pasek AppBar jest inny. Polecenia są domyślnie umieszczone po lewej i prawej stronie ekranu, ponieważ tam najczęściej znajdują się nasze kciuki. Dzięki takiemu rozmieszczeniu poleceń obsługa jest łatwiejsza i zapewnia większy komfort użytkownikowi. Konwencja wyglądu paska aplikacji jest następnym przykładem tego, że platforma aplikacyjna Windows 8 została od początku do końca zbudowana z myślą o dotyku.

Tworzenie przyjaznych w dotyku interakcji za pomocą kontrolki SemanticZoom

Oprócz kontrolki HTML i WinJS, omówionych wcześniej z punktu widzenia przyjazności w dotyku, dostępne są jeszcze dwie inne, o których chciałbym wspomnieć, ponieważ uwydatniają nacisk położony na dotyk w systemie Windows 8. Pierwsza z nich to kontrolka `ListView`, którą widziałeś już wielokrotnie w tej książce.

⁵ Szczegółowe zalecenia firmy Microsoft dotyczące poleceń znajdują się pod adresem <http://msdn.microsoft.com/en-us/library/windows/apps/hh761499.aspx> (<http://tinysells.com/210>).

⁶ Patrz <http://msdn.microsoft.com/en-US/library/windows/apps/hh465415.aspx> (<http://tinysells.com/209>).

Jak już wiesz, kontrolka `ListView` znakomicie nadaje się do wyświetlania danych w formie tabeli lub listy. Jednak być może nie wiesz, że oferuje ona również kilka interakcji dotykowych. Jeżeli np. przesuniesz palcem po elemencie w górę lub w dół, element przemieści się za Twoim palcem i zostanie wybrany, gdy go zwolnisz. Przedstawiam to na rysunku 11.6.

Podobnie jak wszystkich innych elementów opisanych wcześniej, również kontrolki `ListView` możesz używać za darmo w swoich aplikacjach w stylu Windows 8.

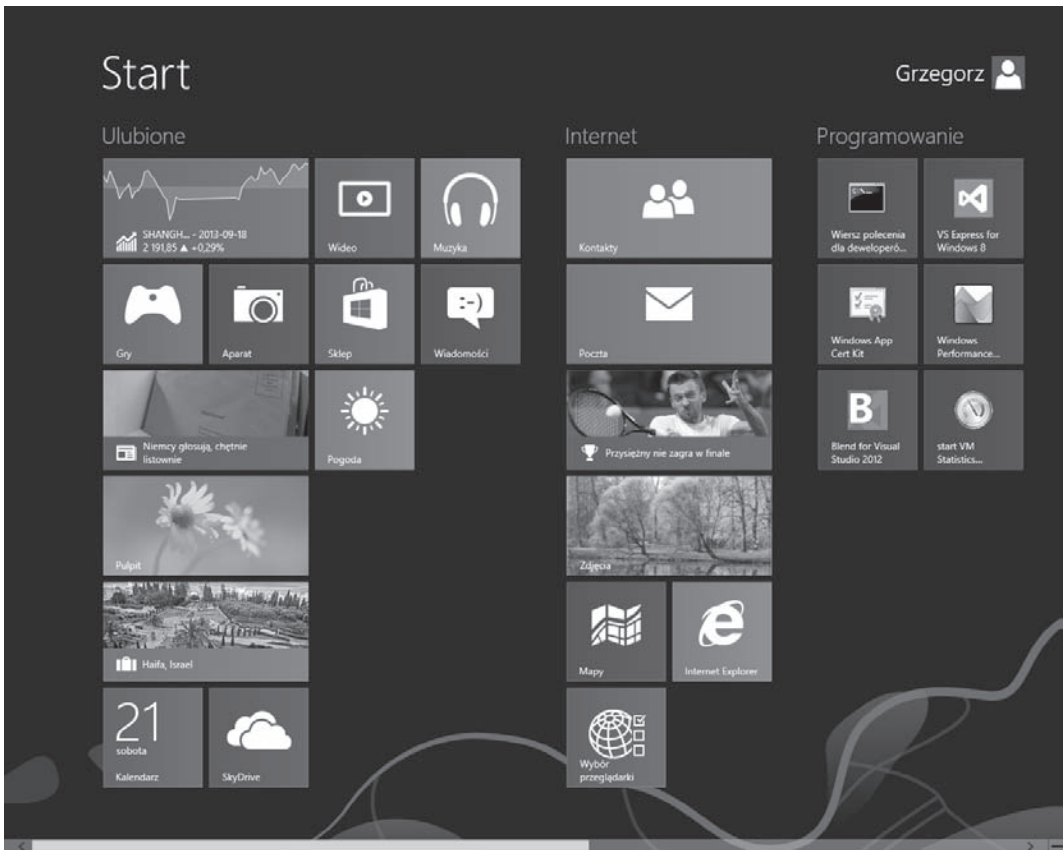


Rysunek 11.6. Wbudowana obsługa dotyku w kontrolce `ListView`

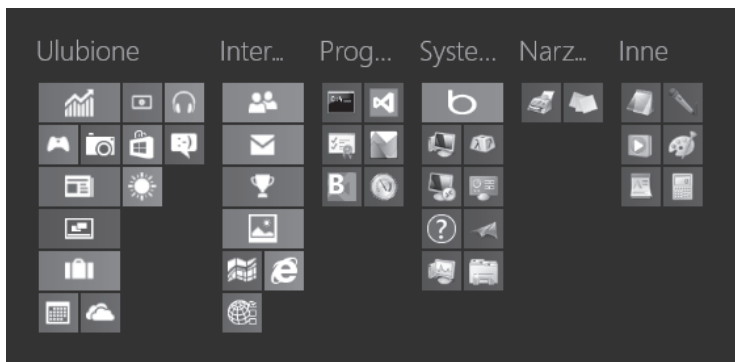
Inną kontrolką, znakomitą dzięki swojej wbudowanej obsłudze dotyku, jest `SemanticZoom`. Jest zbudowana na bazie kontrolki `ListView` i oferuje użytkownikowi, oprócz widoku pełnej listy, również możliwość przełączania pomiędzy wizualizacjami danych, np. według kategorii lub daty. Przypomina to filtrowanie, ale w stylu Windows 8 i z wymyślną nazwą. Najlepszy przykład kontrolki `SemanticZoom` w akcji można zobaczyć na ekranie startowym Windows 8. Domyślnie Twój ekran wygląda prawdopodobnie tak, jak na rysunku 11.7, z kafelkami ułożonymi od lewej do prawej i podzielonymi na grupy.

Ten ekran jest w rzeczywistości kontrolką `SemanticZoom` z pogrupowanymi kontrolkami `ListView` i może być obsługiwany tak, jak każda inna aplikacja, wykorzystująca te kontrolki. A więc, jeżeli zastosujesz powiększenie (przez szczyknięcie, `Ctrl`+kółko myszy lub `Ctrl`+`+`/-), otrzymasz pomniejszony widok, przedstawiający cały ekran startowy z aplikacjami pogrupowanymi według kategorii (patrz rysunek 11.8).

W tym przypadku pomniejszenie umożliwia szybki rzut oka na całą „krajną” aplikacji, dzięki czemu można szybko znaleźć poszukiwany program. Można nadać nazwy grupom aplikacji, przeciągać je, przemieszczać na ekranie i zmieniać ułożenie. Również dodanie kontrolki `SemanticZoom` do własnej aplikacji jest łatwe; spojrzmy na przykład. Załóżmy, że dodaliśmy nowe pole z kategorią w galerii zdjęć z rozdziału 3, „Układ”:



Rysunek 11.7. Ekran startowy Windows 8



Rysunek 11.8. Pomniejszony widok ekranu startowego Windows 8

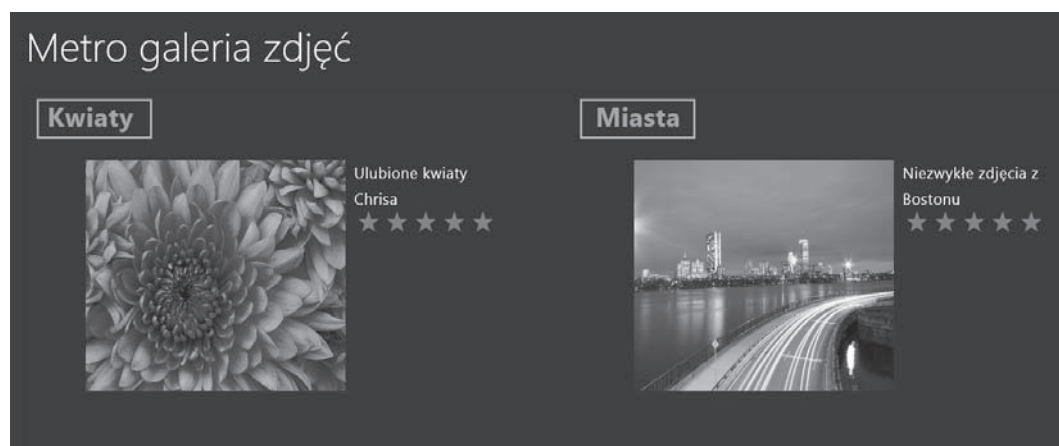
```
var images = new WinJS.Binding.List([
  { url: "images/BostonCityFlow.jpg", caption: "Niezwykłe zdjęcia z Bostonu",
    category: "Miasta" },
  { url: "images/Chrysanthemum.jpg", caption: "Ulubione kwiaty Chrisa",
    category: "Kwiaty" },
  { url: "images/Penguins.jpg", caption: "Pingwiny rozmawiają o Windows 8",
```

```

    category: "Zwierzęta" },
  { url: "images/PensiveParakeet.jpg",
    caption: "Bobby Fischer na szachowych mistrzostwach ptaków", category: "Zwierzęta" },
  { url: "images/CostaRicanFrog.jpg", caption: "Żaba spotkana na Costa Rica",
    category: "Zwierzęta" },
  { url: "images/Jellyfish.jpg", caption: "Czyż meduza nie jest fajna?",
    category: "Zwierzęta" },
  { url: "images/Hydrangeas.jpg", caption: "Ulubione kwiaty Brandona",
    category: "Kwiaty" },
  {url: "images/Koala.jpg", caption: "Cześć!", category: "Zwierzęta" }
]);

```

Teraz, ponieważ dodaliśmy tę kategorię do swoich danych, chcielibyśmy umożliwić użytkownikowi wyświetlenie listy kategorii, oprócz pełnej listy zdjęć. Kontrolką, która oferuje różne tego typu widoki, jest SemanticZoom, ale my, zanim użyjemy swojej ulubionej kontrolki ListView, chcielibyśmy przekształcić ją w pogrupowany widok. Do utworzenia pogrupowanych kontrollek ListView możemy zastosować dokładnie ten sam sposób, jaki został opisany w rozdziale 2. Rezultat przedstawiam na rysunku 11.9.



Rysunek 11.9. Pogrupowane kontrolki ListView

Pogrupowaliśmy już zdjęcia według kategorii, musimy zatem dopisać kilka małych dodatków, aby kontrolka SemanticZoom zaczęła działać w naszej aplikacji. Przede wszystkim dodamy inny szablon elementów reprezentujących pomniejszone elementy:

```

<div id="zoomTemplate" data-win-control="WinJS.Binding.Template">
  <div class="zoomItem">
    <h1 class="zoomItem-Text" data-win-bind="innerText: title"></h1>
  </div>
</div>

```

Pomniejszony widok składa się po prostu z tytułów kategorii, z kilkoma stylami CSS powiększającymi czonkę i umieszczającymi każdy tytuł na eleganckim tle w stylu Windows 8. Po zbudowaniu takiego szablonu utworzymy kontrolkę SemanticZoom (zauważ, że pierwszy znacznik <div> musi określać widok powiększony, a drugi pomniejszony):

```

<div id="mainContainer" data-win-control="WinJS.UI.SemanticZoom">
  <!-- Widok powiększony -->
  <div id="imgContainer" data-win-control="WinJS.UI.ListView"

```

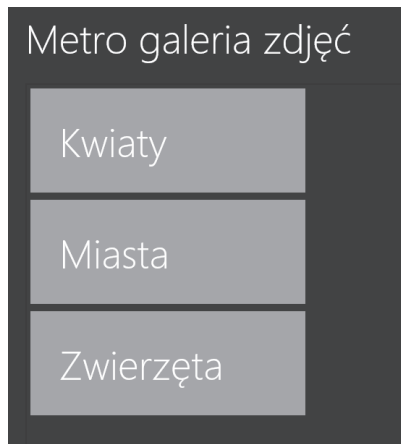
```

data-win-options="{ itemDataSource: Photos.groupedImages.dataSource,
                    itemTemplate: select('#imgTemplate'),
                    groupDataSource:
                    Photos.groupedImages.groups.dataSource,
                    groupHeaderTemplate:
                    select('#headerTemplate'),
                    layout: {type: WinJS.UI.GridLayout} }">
</div>
<!-- Widok pomniejszony. -->
<div id="categoryContainer" data-win-control="WinJS.UI.ListView"
    data-win-options="{ itemDataSource:
                    Photos.groupedImages.groups.dataSource,
                    itemTemplate:
                    select('#zoomTemplate'),
                    selectionMode: 'none',
                    tapBehavior: 'invoke' }">

</div>
</div>

```

Z perspektywy znaczników kontrolka SemanticZoom nie jest niczym innym, jak tylko tagiem `<div>` obejmującym każdy „poziom powiększenia” naszych danych, w tym przypadku kategorii i obrazów. Magia kontrolki ujawnia się podczas działania programu. Zachęcam do wypróbowania jej i uruchomienia aplikacji ze źródłem danych online. Po uruchomieniu aplikacji zobaczysz listę kategorii, pokazaną na rysunku 11.10.



Rysunek 11.10. Widok danych w kontrolce SemanticZoom

Tutaj właśnie ujawniają się ciekawe, przyjazne w dotyku elementy. Przede wszystkim, kiedy dotkniesz lub klikniesz jedną z kategorii w wspomnianej wcześniej liście, widok automatycznie przełączy się na listę obrazów i przejdziesz do grupy, którą kliknąłeś. Jeżeli np. klikniesz *Kwiaty*, zostaniesz od razu przeniesiony do sekcji z kwiatami zawierającej pełną listę obrazów. Co ciekawe, dodatkowo jest jeszcze wbudowana obsługa powiększania przez szczypanie. Standardowo kontrolka SemanticZoom obsługuje szczypanie i rozciąganie (szczypanie pomniejsza, rozciąganie powiększa) oraz przenosi pomiędzy poziomami powiększenia. Możesz powiększyć główny widok przedstawiony na rysunku 11.9 i wyświetlić główną listę, szczypiąc gdziekolwiek na ekranie, i z powrotem go pomniejszyć, rozciągając palcami. Jest to rewelacyjna interakcja dotykowa pozostająca do Twojej dyspozycji bez dodatkowych kosztów.

Interakcje za pomocą myszy i klawiatury

W tym rozdziale mówiłem już dużo o dotyku, o tym, że system Windows 8 został zbudowany z myślą o nim, oraz jak powinieneś tworzyć aplikacje, mając na względzie przede wszystkim dotyk. Zanim przejdziemy do głębszej dyskusji na temat urządzeń, chciałbym wyraźnie podkreślić, że dotyk przede wszystkim to nie znaczy wyłącznie dotyk. Inne systemy faworyzują albo dotyk, albo klikanie, z uszczerbkiem dla jednego lub drugiego, ale Windows 8 do nich nie należy. Każda dotykowa interakcja z platformą na swój odpowiednik dla myszy lub klawiatury, prosty i w większości przypadków intuicyjny. Przykładowo aktywacja paska aplikacji — w przypadku dotyku jest to przesunięcie palcem do góry ekranu — polega po prostu na kliknięciu prawym przyciskiem myszy w oknie aplikacji lub naciśnięciu klawiszy *Win+Z*. Aby uaktywnić boczny pasek menu, należy nacisnąć *Win+C* lub przesunąć kursor do dolnego lub górnego prawego rogu ekranu.

Ta sama zasada obowiązuje w przypadku kontrolki *ListView* oraz *SemanticZoom*. Aby wybrać element w liście *ListView* (gest stuknięcia), kliknij go prawym przyciskiem lub użyj klawiszy strzałek i naciśnij *Ctrl+Enter*. Aby skorzystać z kontrolki *SemanticZoom*, możesz nacisnąć klawisz *Ctrl* i obrócić kółkiem myszy lub nacisnąć klawisze *Ctrl++* albo *Ctrl+-* w taki sam sposób, jak przy powiększaniu i pomniejszaniu stron internetowych w przeglądarce.

Oprócz wykorzystania funkcjonalności platformy realizujących tradycyjną interakcję za pomocą klawiatury lub myszy, Microsoft zaleca dalej stosować interakcje używane przez strony internetowe i aplikacje od zawsze. Niektóre z nich to podpowiedzi, klawisze dostępu i zakładki. Jeżeli jesteś programistą stron internetowych, na pewno dobrze znasz te metody, a jeżeli nie, to na stronie Windows DevCenter znajdziesz mnóstwo przydatnych informacji zarówno o wyżej wymienionych, jak i innych interakcjach za pomocą myszy lub klawiatury w aplikacjach w stylu Windows 8⁷.

System Windows 8 łączy tradycyjne metody interakcji z użytkownikiem, takie jak klawiatura lub mysz, z nową wizją, jaką jest dotyk. Jednak zamiast wykorzystać dotyk jako pretekst do zrezygnowania z myszy i klawiatury, Windows 8 przywiązuje do obu form interakcji jednakową wagę. Dla Ciebie, jako programisty, oznacza to, że możesz tworzyć wspaniałe aplikacje w stylu Windows 8, ale wciąż jednak umożliwiać użytkownikom komputerów stacjonarnych korzystanie z efektów Twoich wysiłków i obsługiwanie komputerów w taki sposób, jaki im najbardziej odpowiada. Jest to podejście niewymagające kompromisów, stosowane przez Microsoft przy każdej okazji.

Dotychczas w tym rozdziale wiele mówiłem o dotyku i sposobach interakcji użytkowników z ich urządzeniami. Teraz skupię się na interakcji urządzeń z użytkownikami i otaczającym je światem oraz na tym, jak my, programiści, możemy tworzyć ciekawe aplikacje w stylu Windows 8, wykorzystujące coraz większą liczbę czujników i różnych funkcjonalności zawartych w nowoczesnych urządzeniach.

Wykorzystanie możliwości urządzenia

Dzisiejsze urządzenie przenośne jest czymś więcej niż tabliczką z dotykowym ekranem i programową klawiaturą. Wiele współczesnych urządzeń posiada coraz większy wachlarz niezwykłych urządzeń peryferyjnych, czujników i funkcjonalności umożliwiających — i urządzeniu, i programiście — wykonywanie czynności, które zaledwie kilka lat temu wydawały się być poza zasięgiem. W tej części rozdziału omówię kilka popularnych i tradycyjnych cech urządzenia, takich jak nagrywanie dźwięku lub drukowanie, a w kolejnych częściach zajmiemy się dokładniej nowoczesnymi czujnikami i funkcjonalnościami.

⁷ Patrz <http://msdn.microsoft.com/en-US/library/windows/apps/hh465415.aspx> (<http://tinysells.com/209>).

Deklarowanie możliwości urządzenia

Przed wszystkim przypomnijmy sobie krótko coś, co już znamy, czyli plik manifestu *package.appxmanifest*. Jak już widziałeś wiele razy w tej książce, plik ten jest podstawowym źródłem metadanych dotyczących Twojej aplikacji, poczynwszy od nazwy pakietu i numeru wersji, a skończywszy na wskaźnikach do plików z obrazami, reprezentującymi różne logo prezentowane użytkownikom. Plik ten zawiera również bardzo ważne zakładki *Możliwości* i *Deklaracje*, informujące system Windows 8 o potrzebie dostępu Twojej aplikacji do możliwości urządzenia lub wykonania ważnych zadań w tle nawet wtedy, kiedy aplikacja nie jest uruchomiona. Jeśli wykorzystujesz funkcjonalności urządzenia opisane w tym i innych rozdziałach, nie zapomnij o sprawdzeniu, czy w zakładce *Możliwości* znajduje się odpowiednia deklaracja.

Jeżeli miałeś już do czynienia z którąś z funkcjonalności i otrzymałeś komunikat „Access Denied” (odmowa dostępu), prawdopodobnie powinieneś zadeklarować odpowiednią możliwość. Dlatego sprawdź najpierw zawartość zakładki *Możliwości*. W różnych przykładach w tym rozdziale będziemy ustawiać takie deklaracje jak *Lokalizacja* czy *Mikrofon*.

Praca z urządzeniami rejestrującymi

W rozdziale 5., „Media”, dowiedziałeś się, jak uzyskać dostęp do kamery użytkownika, aby zrobić zdjęcie lub nagrać film. Jak się zapewne domyślasz, możliwy jest również dostęp do mikrofonu, by emitować lub nagrywać dźwięk. Aby to osiągnąć, należy najpierw zaznaczyć opcję *Mikrofon* w pliku *package.appxmanifest*, a następnie można dodać kilka przycisków do rozpoczęcia i zakończenia nagrywania, tag span do pokazywania statusu oraz audio, umożliwiający użytkownikowi odtworzenie nagrania:

```
<p><audio id="audioTarget" controls></audio></p>
<p><span id="status"></span></p>
<p>
  <button id="record">Rejestruj dźwięk</button>
  <button id="stop">Zatrzymaj rejestrowanie</button>
</p>
```

Następnie należy dodać logikę do metody *ready*, aby wyszukać wszystkie dostępne urządzenia rejestrujące:

```
var deviceEnum = Windows.Devices.Enumeration;
var deviceInfo = deviceEnum.DeviceInformation;

deviceInfo.findAllAsync(deviceEnum.DeviceClass.audioCapture)
.done(function (devices) {
  var deviceList = devices;
});
```

Nie jest rzadkością posiadanie przez użytkownika komputera stacjonarnego kilku kamer i mikrofonów. System Windows oferuje przestrzeń *Windows.Devices.DeviceInformation* do wyliczenia urządzeń dostępnych dla użytkownika. Metoda *findAllAsync* zapytuje system operacyjny o listę dostępnych urządzeń, opartych w tym przypadku na klasie *DeviceClass.audioCapture*, i zwraca listę, którą można następnie przedstawić użytkownikowi, dzięki czemu będzie mógł wybrać preferowane urządzenie rejestrujące.

W tym przykładzie zamierzamy przypisać pierwsze urządzenie do obiektu *MediaCaptureInitializationSettings* ➔ *Settings()*:

```
var capture = Windows.Media.Capture;
var captureSettings = new capture.MediaCaptureInitializationSettings();
```

```
captureSettings.audioDeviceId = deviceList[0].id;
captureSettings.streamingCaptureMode = capture.StreamingCaptureMode.audio;
```

Oprócz określenia urządzenia wybranego do rejestracji, musimy również wyspecyfikować dźwięk jako tryb rejestracji. Po utworzeniu obiektu z ustawieniami początkowymi można przygotować obiekt `MediaCapture` do rejestracji:

```
var profile;
var mediaProperties = Windows.Media.MediaProperties;
mediaCapture = new Windows.Media.Capture.MediaCapture();
mediaCapture.initializeAsync(captureSettings).done(function (result) {
    profile = mediaProperties.MediaEncodingProfile.createMp3(
        mediaProperties.AudioEncodingQuality.high);
});
```

Po zainicjowaniu obiektu `MediaCapture` za pomocą obiektu `captureSettings` utworzymy profil kodowania dźwięku generującego wysokiej jakości zapis MP3 wszystkiego, co tylko użytkownik powie do mikrofonu. Po ukończeniu wszystkich ustawień jesteśmy gotowi do podłączenia kilku uchwytów zdarzeń do przycisków start i stop:

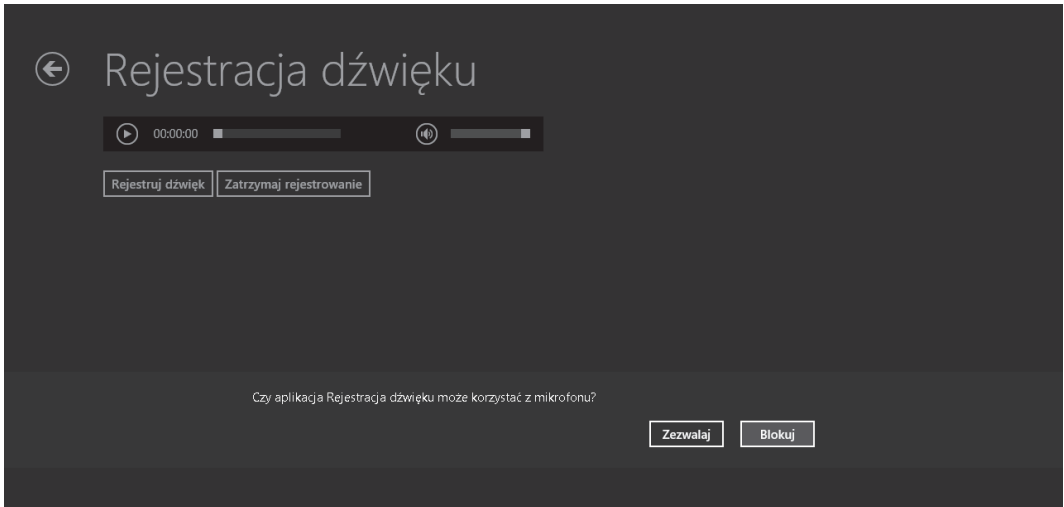
```
document.querySelector('#record').addEventListener('click', startRecording);
function startRecording () {
    var storage = Windows.Storage;
    storage.KnownFolders.musicLibrary.createFileAsync('audioCapture.mp3',
        storage.CreationCollisionOption.replaceExisting).then(function (file) {
        audioFile = file;
        return mediaCapture.startRecordToStorageFileAsync(profile, audioFile)
            .done(
                function (result) {
                    document.querySelector('#status').innerText =
                        "Rejestracja rozpoczęta...";
                });
    });
}

document.querySelector('#stop').addEventListener('click', stopRecording);
function stopRecording () {
    mediaCapture.stopRecordAsync().done(function (result) {
        var file = window.URL.createObjectURL(audioFile);
        document.querySelector('#audioTarget').src = file;

        document.querySelector('#status').innerText =
            "Rejestracja zakończona...";
    });
}
```

Kiedy użytkownik naciśnie przycisk start, zostanie utworzony nowy plik MP3 dla nagrania i rozpocznie się rejestracja dźwięku. Kiedy użytkownik zakończy rejestrację, zostanie utworzony adres `ObjectURL` reprezentujący utworzony plik i ten adres zostanie przypisany do elementu audio naszej aplikacji. Kiedy dodasz skrypt JavaScript z powyższego kodu (i upewnij się, że zadeklarowana jest możliwość *Mikrofon*), wówczas po uruchomieniu aplikacji otrzymasz komunikat podobny do pokazanego na rysunku 11.11.

Gdy mamy do dyspozycji takie możliwości jak audio, wideo lub geolokalizacja, zadeklarowanie potrzeby danej możliwości jest tylko połową sukcesu. Użytkownik musi pozwolić Twojej aplikacji na użycie tych urządzeń w systemie. W tym przypadku rejestracja dźwięku jest możliwa tylko wtedy, gdy użytkownik kliknie przycisk *Zezwalaj*. Jeżeli po kliknięciu tego przycisku, a następnie *Rejestruj dźwięk*, zaśpiewasz



Rysunek 11.11. Aplikacja w stylu Windows 8 pytająca użytkownika o zezwolenie na dostęp do urządzenia systemowego

kilka taktów ulubionej piosenki i klikniesz przycisk *Zatrzymaj rejestrowanie*, aplikacja zakończy nagrywanie w obiekcie `MediaCapture`, pobierze obiekt `audioFile` zawierający nagranie w formacie modułu dźwiękowego i przypisze adres `ObjectUrl` do elementu `audio`:

```
mediaCapture.stopRecordAsync().done(function (result) {
    var file = window.URL.createObjectURL(audioFile);
    document.querySelector('#audioTarget').src = file;

    document.querySelector('#status').innerText = "Rejestracja zakończona...";
}, function (error) {
    document.querySelector('#status').innerText = error.msg;
});
```

Po wykonaniu powyższych czynności można kliknąć przycisk odtwarzania umieszczony w tagu `audio` i odsłuchać swój zdumiewająco zwyczajny głos płynący z głośników komputera.

Dodawanie możliwości drukowania

Ostatnią możliwością urządzenia, którą omówię w tym punkcie, jest stare, dobre drukowanie. System oferuje pełną obsługę drukowania w aplikacjach w stylu Windows 8. Dodanie jej do aplikacji jest całkiem proste, nawet z dostosowaniem, które elementy mają być widoczne w widoku wydruku.

Drukowanie jest jedną z możliwości, która nie wymaga deklaracji w pliku *package.appxmanifest*, możemy więc od razu zagłębić się w kod. W tym przykładzie chcemy dodać do aplikacji RSS Reader, utworzonej w rozdziale 1., „Witaj, Windows 8”, przycisk *Drukuj wpis*. Zaczniemy od dodania przycisku *Drukuj wpis* na stronie *posts.html*:

```
<h1 class="titlearea win-type-ellipsis">
  <span class="pagetitle"></span> <!--Ustaw tytuł blogu -->
  <button id="print">Drukuj wpis</button>
</h1>
```

Następnie, gdy aplikacja będzie uruchamiana, zainicjalizujemy obiekt `PrintManager`:

```
function initPrint() {
    var printManager =
        Windows.Graphics.Printing.PrintManager.getForCurrentView();

    printManager.addEventListener('printtaskrequested',
    function print (printEvent) {
        printEvent.request.createPrintTask("Drukuj blog", function (args) {
            args.setSource(MSApp.getHtmlPrintDocumentSource(document));
        });
    });
}
```

Rozpoczynamy od wywołania metody `getForCurrentView` z obiektu `Windows.Graphics.Printing.PrintManager`. Następnie tworzymy nasłuch zdarzenia `'printtaskrequested'`, a kiedy się ono pojawi, tworzymy zadanie drukowania odczytujące stronę (obiekt zawierający dokument HTML) przeznaczoną do wydruku. Mając wszystko przygotowane, możemy skonfigurować przycisk *Drukuj wpis*:

```
document.getElementById('print').addEventListener('click', function () {
    Windows.Graphics.Printing.PrintManager.showPrintUIAsync();
});
```

Szczegółowe wyjaśnienia nie są tutaj potrzebne. Kiedy użytkownik kliknie przycisk *Drukuj wpis*, pojawi się strona wydruku w stylu Windows 8, przedstawiona na rysunku 11.12.



Rysunek 11.12. Drukowanie w aplikacji w stylu Windows 8

Mimo że całość wygląda bardzo ładnie, nie jesteśmy zbyt zadowoleni z tego, że przyciski wstecz i *Drukuj wpis* są widoczne na podglądzie wydruku. Okazuje się jednak, że stary znajomy z rozdziału 3., czyli zapytanie o media, szybko pomoże oczyścić widok wydruku naszego dokumentu:⁸

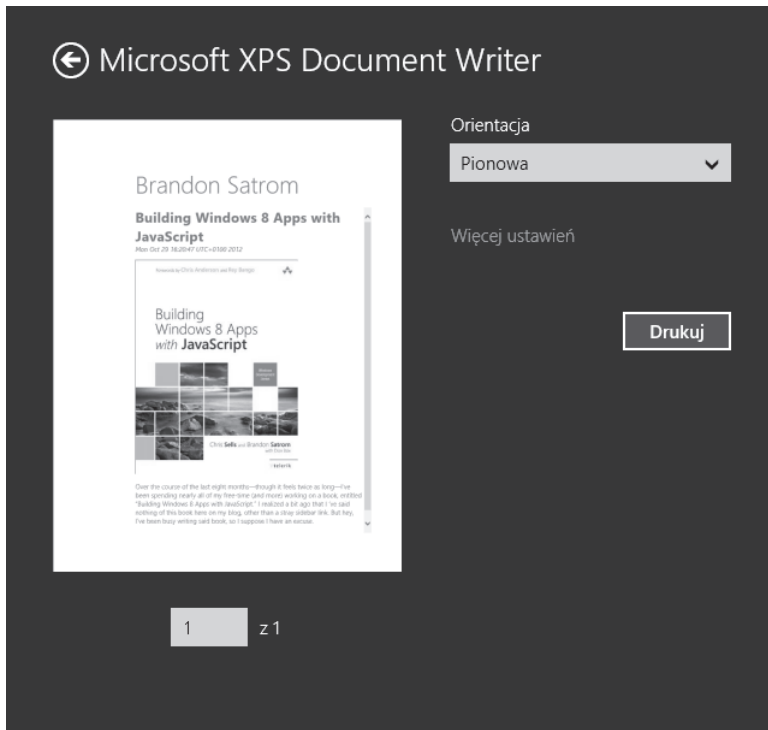
⁸ Więcej informacji na temat tworzenia stylów CSS do drukowania znajduje się pod adresem <http://benfrain.com/create-print-styles-using-css3-media-queries/>.


```
@media print {
  .noprint {
    display:none;
  }
}
```

W przypadku tego konkretnego zapytania o media, dotyczącego tylko dokumentu w widoku wydruku, utworzymy klasę `noprint`, która ukryje każdy element, w jakim się ją zastosuje. Tu ukryjemy w wydruku nasze przyciski:

```
<button class="win-backbutton noprint" aria-label="Wstecz" disabled></button>
<h1 class="titlearea win-type-ellipsis">
  <span class="pagetitle"> </span>
  <button id="print" class="noprint">Drukuj wpis</button>
</h1>
```

Wszystko jest bardzo proste i teraz otrzymujemy przyjazny dla drukarki widok wydruku wpisu, przedstawiony na rysunku 11.13.



Rysunek 11.13. Zastosowanie zapytania o media do utworzenia widoku przyjaznego dla drukarki

Dotychczas napisałem bardzo dużo na temat podstaw interakcji z urządzeniem w aplikacjach w stylu Windows 8, ale tak naprawdę zaledwie musnąłem temat. W kolejnych dwóch podrozdziałach opiszę więcej nowoczesnych możliwości systemu Windows 8. Teraz kolej na integrację w aplikacji obsługi lokalizacji geograficznej.

Praca z danymi lokalizacyjnymi

Jeżeli posiadasz smartfon, na pewno korzystałeś z aplikacji lub wbudowanej funkcjonalności udzielającej wskazówek, jak trafić do zadanego celu z miejsca, w którym się znajdujesz, albo gdzie znaleźć najbliższą restaurację, hotel lub stację benzynową w nieznanej okolicy. Technicznym określeniem tej funkcjonalności jest geolokalizacja; opisuje ona cechę urządzenia, pozwala wykorzystać zewnętrzne dane lokalizacyjne (system GPS lub triangulację na podstawie komórkowych stacji bazowych i danych WiFi) do określenia miejsca, w którym się znajdujesz, i przekazuje te informacje aplikacji lub usłudze, która ich zażąda. Twoja lokalizacja jest zazwyczaj wyrażona za pomocą danych, takich jak długość i szerokość geograficzna, kierunek, prędkość itp. Po uzyskaniu tej informacji urządzenie może oferować takie funkcjonalności jak nawigacja do celu krok po kroku lub najbliższa budka z hot dogami otwierana o drugiej nad ranem.

Użycie obiektu Geolocator

Funkcjonalność geolokalizacji jest dostępna dla programistów w interfejsie API Geolocation języka HTML5 (`window.navigator.geolocation`) lub w przestrzeni `Windows.Devices.Geolocation`. Ta druga możliwość funkcjonuje w aplikacjach w stylu Windows 8 tak samo jak w przeglądarce Internet Explorer lub innych, dlatego tu nie będę jej opisywać. Natomiast przy migracji aplikacji webowej, wykorzystującej usługi lokalizacyjne, do systemu Windows 8 warto wiedzieć, że interfejs API Geolocation HTML5 też jest w pełni obsługiwany. A teraz spójrzmy na obiekt `Geolocator` środowiska WinRT:

```
var locator = new Windows.Devices.Geolocation.Geolocator();
locator.getGeopositionAsync().done(getPositionHandler, errorHandler);
```

```
function getPositionHandler(location) {
    document.getElementById("latitude").innerHTML =
        location.coordinate.latitude;
    document.getElementById("longitude").innerHTML =
        location.coordinate.longitude;
    document.getElementById("accuracy").innerHTML =
        location.coordinate.accuracy;
    document.getElementById("status").innerHTML =
        statusMsg(locator.locationStatus);
}
function errorHandler(err) {
    document.getElementById("status").innerHTML =
        statusMsg(locator.locationStatus);
}

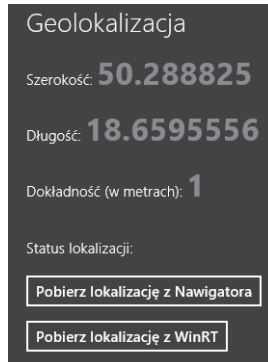
function statusMsg(locStatus) {
    switch (locStatus) {
        case Windows.Devices.Geolocation.PositionStatus.ready:
            return "Lokalizacja dostępna";
        case Windows.Devices.Geolocation.PositionStatus.initializing:
            return "Urządzenie GPS w trakcie inicjalizacji";
        case Windows.Devices.Geolocation.PositionStatus.noData:
            return "Dane z usług lokalizacyjnych niedostępne";
        case Windows.Devices.Geolocation.PositionStatus.disabled:
            return "Usługi lokalizacyjne w trakcie ustalania pozycji";
        case Windows.Devices.Geolocation.PositionStatus.notInitialized:
            return "Aplikacja nie żądała danych lokalizacyjnych";
        case Windows.Devices.Geolocation.PositionStatus.notAvailable:
```

```

    return "Brak w systemie żądanej usługi lokalizacyjnej";
  default:
    break;
}
}

```

Po utworzeniu instancji obiektu `Geolocator` wywołujemy metodę `getPositionAsync` inicjalizującą zapytanie lokalizacyjne. Po jej wykonaniu aplikacja wywołuje jedną z procedur obsługi i wyświetla rezultaty na ekranie w sposób pokazany na rysunku 11.14.



Rysunek 11.14. Określanie pozycji użytkownika za pomocą obiektu nawigacji w JavaScriptcie

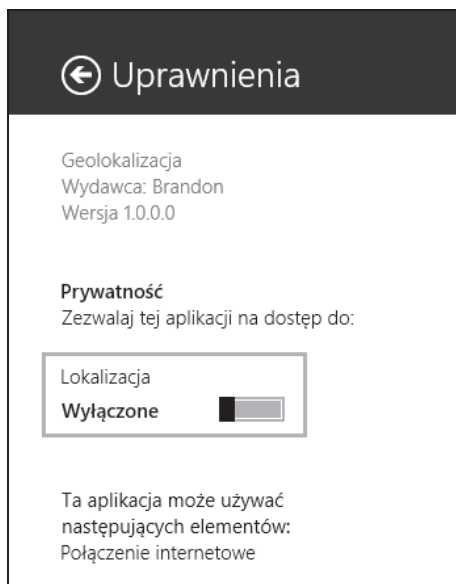
Obiekt `location` zwrócony przez konstruktor `Windows.Devices.Geolocator` zawiera te same dane dotyczące długości, szerokości geograficznej i dokładności, jak obiekt nawigacji w przeglądarce, ale zawiera również właściwość `locationStatus`, której tam nie ma. Wartość `locationStatus` dostarcza nieco więcej informacji o bieżącym statusie żądania pozycji, a wartości, jakie może przyjąć, można znaleźć w enumeracji `Windows.Devices.Geolocation.PositionStatus`. Jak zapewne zauważyłeś, w metodzie `statusMsg` enumeracja `PositionStatus` zawiera zarówno kody normalnego wykonania żądania, jak i kody błędów, które są wykorzystywane do wywołania i procedur obsługi błędów, i pomyślnego wykonania.

Kiedy wystąpi błąd, mogą być trzy przyczyny nieudanego określenia pozycji użytkownika. Przede wszystkim użytkownik może wyłączyć ujawnianie swojej lokalizacji za pomocą komunikatu `MessageDialog`, wyświetlanego przy pierwszym żądaniu, lub w oknie *Uprawnienia* w pasku *Ustawienia*, przedstawionym na rysunku 11.15.

Inne dwie możliwe przyczyny nieudanego zlokalizowania użytkownika to przekroczenie czasu sieci lub bardziej ogólny problem „nieдоступności pozycji”, którego powód może być niemal dowolny. Niezależnie od tego, Twoja aplikacja powinna zawsze obsługiwać błędy lokalizacyjne i poprawnie kończyć działanie nawet wtedy, kiedy musisz nakłaniać użytkownika do włączenia odpowiedniej usługi, wymaganej przez Twoją aplikację.

Jeżeli w tej samej aplikacji wykorzystasz jednocześnie dwie metody geolokalizacji, bardzo prawdopodobne jest, że obie dostarczą dokładnie te same wyniki. Obie metody, mimo iż mają inne nazwy, w celu uzyskania danych lokalizacyjnych wywołują te same usługi systemu operacyjnego. Tak więc decyzja, której metody użyć, jest w rzeczywistości kwestią gustu i osobistych upodobań.

W tym miejscu możesz zastanawiać się, jak naprawdę i kiedy działają usługi lokalizacyjne systemu Windows 8. Zapewne oczekujesz, że pracują one wtedy, gdy w urządzeniu zostanie wykryty moduł GPS. Ale zdziwisz się, gdy zobaczysz, że usługi lokalizacyjne są dostępne nawet wówczas, gdy takie urządzenie nie jest dostępne w systemie. Dotyczy to również przeglądarek obsługujących język HTML5.



Rysunek 11.15. Ustawienia uprawnień w pasku Uprawnienia

W praktyce wszystkie przeglądarki podejmują jedną lub dwie akcje, gdy aplikacja zażąda dostępu do usług lokalizacyjnych. Wykorzystują wtedy wbudowany moduł GPS lub określają pozycję użytkownika na podstawie jego adresu IP i usług mapujących. Pierwsza metoda jest o wiele dokładniejsza i preferowana, jeżeli jest dostępna, ale dzięki drugiej z nich urządzenia bez modułu GPS (jak większość komputerów stacjonarnych i laptopów) mogą korzystać z usług lokalizacyjnych. Oczywiście, nie jest to sposób uzyskania wiarygodnych i precyzyjnych danych, ale bardzo wygodny przy wyszukiwaniu restauracji lub firm w najbliższej okolicy.

Obserwacja zmian położenia

Omówione dotychczas przykłady lokalizacji dotyczą pojedynczego żądania określenia pozycji, ponieważ dostarczają jeden wynik lokalizacyjny. Ten sposób jest przydatny w wielu przypadkach, ale w niektórych aplikacjach — np. nawigacyjnych — wymagany jest dostęp do ciągłych aktualizacji położenia. Zarówno nawigator JavaScript, jak i obiekt `Geolocation` środowiska WinRT oferują dostęp do regularnych aktualizacji. Do obiektu `Geolocation` musisz jedynie dodać nasłuch zdarzenia `'positionchanged'`:

```
locator.addEventListener('positionchanged', getPositionHandler);
```

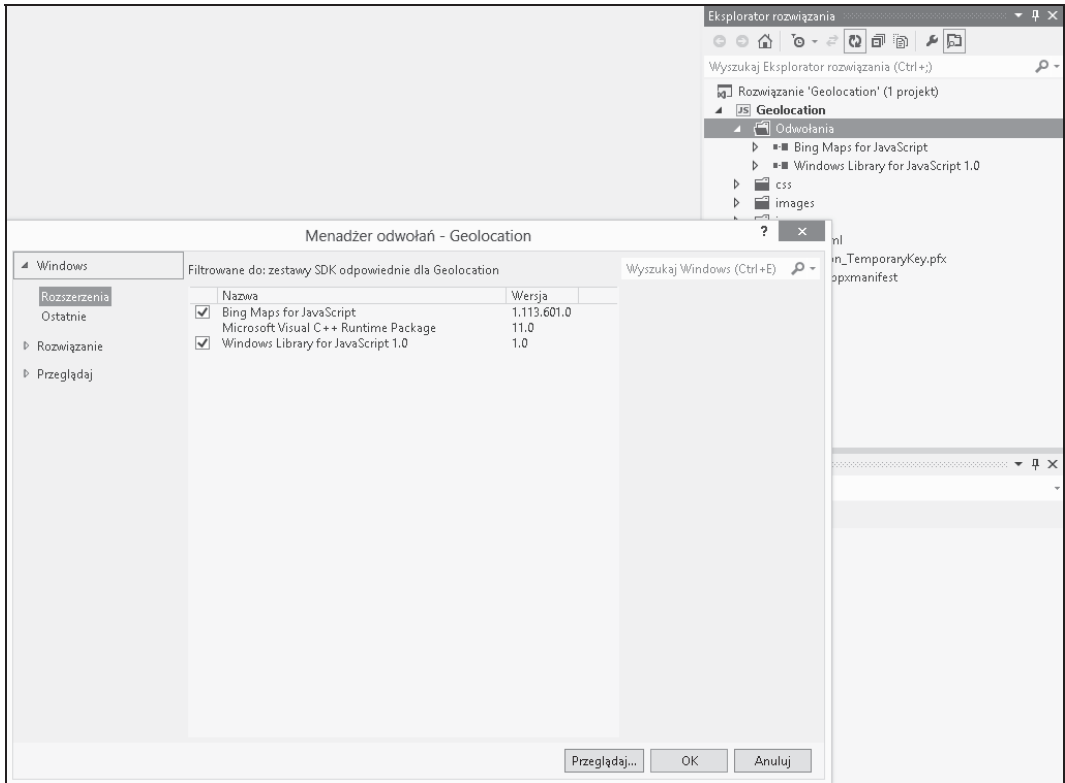
Tak jak we wszystkich procedurach obsługi zdarzeń, jeżeli nie potrzebujesz danych lokalizacyjnych, możesz usunąć nasłuch:

```
locator.removeEventListener('positionchanged', getPositionHandler);
```

Małe ostrzeżenie dotyczące tego sposobu: ponieważ zmiany lokalizacji są aktywnie monitorowane poprzez sieć lub GPS, dlatego może to być operacja intensywnie wykorzystująca baterię.

Użycie danych lokalizacyjnych z mapami Bing Maps

Długość i szerokość geograficzna nie są same w sobie zbyt pasjonujące. Rozbudujmy więc nasz przykład o użycie danych lokalizacyjnych w systemie Windows 8 do określenia położenia użytkownika na mapie. W tym przykładzie zamierzam użyć pakietu Bing Maps SDK dla aplikacji w stylu Windows 8 (wersja Release), który możesz pobrać z galerii Visual Studio⁹. Po pobraniu i zainstalowaniu pakietu należy dodać do aplikacji odwołanie, klikając prawym przyciskiem myszy folder *Odwołania* w projekcie Microsoft Visual Studio i wybierając *Dodaj odwołanie*. Pojawi się okno *Menadżer odwołań*, pokazane na rysunku 11.16.



Rysunek 11.16. Dodanie odwołania do pakietu SDK w środowisku Visual Studio 2012

Po zaznaczeniu pozycji *Bing Maps for JavaScript (RP)* możemy kliknąć *OK* i powrócić do naszej aplikacji. Teraz możemy dodać mapę Bing. Najpierw musimy umieścić na stronie kilka odwołań do skryptów oraz tag `<div>` dla mapy:

```
<script type="text/javascript" src=
  "/Bing.Maps.JavaScript/js/veapicore.js"></script>
<script type="text/javascript" src=
  "/Bing.Maps.JavaScript/js/veapiModules.js"></script>
```

⁹ Dostępny pod adresem <http://visualstudiogallery.msdn.microsoft.com/ebc98390-5320-4088-a2b5-8f276e4530f9> (<http://tinysells.com/212>).

```
<div id="mapContainer">
  <div id="map"></div>
</div>
```

Następnie inicjalizujemy wszystko w momencie uruchamiania aplikacji:

```
Microsoft.Maps.loadModule('Microsoft.Maps.Map', { callback: initMap });
function initMap() {
  var mapOptions = {
    credentials: "TwojePoświadczenia",
    center: new Microsoft.Maps.Location(40.71, -74.00),
    mapTypeId: Microsoft.Maps.MapTypeId.road,
    zoom: 8
  };
  map = new Microsoft.Maps.Map(document.getElementById("map"), mapOptions);
}
```

Po zakończeniu inicjalizacji ostatnim krokiem jest utworzenie funkcji, którą nasza aplikacja będzie wywoływać po uzyskaniu położenia użytkownika:

```
function addToMap(lat, long) {
  map.setView({
    center: new Microsoft.Maps.Location(lat, long),
    mapTypeId: Microsoft.Maps.MapTypeId.auto, zoom: 18
  });

  var pushpin = new Microsoft.Maps.Pushpin(map.getCenter(), null);
  map.entities.push(pushpin);
}
```

Mając długość i szerokość geograficzną, można ustawić środek mapy i dodać szpilkę pokazującą dokładną lokalizację. Na rysunku 11.17 przedstawiam końcowy rezultat.

Symulowanie informacji o lokalizacji

Podczas pracy z usługami lokalizacyjnymi często musimy przetestować większą liczbę lokalizacji, oprócz położenia domu lub miejsca pracy. Na szczęście, system Windows 8 oferuje funkcjonalność symulacji lokalizacji, dostępną za pomocą opcji *Ustaw lokalizację* w menu symulatora, przedstawionego na rysunku 11.18.

Kliknięcie tej opcji otwiera okno dialogowe, w którym można określić długość i szerokość geograficzną. Dane te będą wysyłane do naszej aplikacji po każdym żądaniu lokalizacyjnym. Jest to opcja bardzo przydatna podczas testowania usług lokalizacyjnych.

Teraz, po omówieniu usług lokalizacyjnych, podsumujemy naszą dyskusję o możliwościach urządzenia przeglądem dodatkowych czujników i interfejsów API, dostępnych dla programistów aplikacji w stylu Windows 8.

Praca z czujnikami

Praca z danymi o lokalizacji użytkownika jest często wykorzystywanym i prezentowanym przykładem możliwości nowoczesnego urządzenia, ale to tylko jeden aspekt z szerokiego zestawu funkcjonalności, do których programiści mają dostęp w systemie Windows 8. W ostatnim podrozdziale tego rozdziału



Rysunek 11.17. Zastosowanie usług lokalizacyjnych z pakietem Bing Maps SDK

opiszę dokładniej przestrzeń `Windows.Devices.Sensors`, zawierającą mnóstwo interfejsów API do różnych urządzeń, które można wykorzystywać na różne sposoby. W tabeli 11.1 zawarłem podsumowanie tych czujników i oferowanych przez nie możliwości.

W tej książce nie ma miejsca, aby omówić wszystkie czujniki, ale te, które opiszę, dadzą Ci nie tylko wyobrażenie o użyteczności czujników w systemie Windows 8, ale również o sposobie ich wykorzystania, wspólnym dla wszystkich czujników. Mówiąc prościej, gdy pokażę, jak używać jednego czujnika, będziesz wiedział większość tego, co będzie Ci potrzebne do pracy z wszystkimi.

Zanim jednak przejdę do szczegółów, napiszę o testowaniu aplikacji wykorzystujących te czujniki. Z wyjątkiem prostego czujnika położenia i w odróżnieniu od funkcjonalności lokalizacyjnych systemu Windows 8, nie ma możliwości przetestowania czujników ani ich działania, jeżeli nie są fizycznie dostępne w urządzeniu. Symulator Windows 8 nie ma żadnej możliwości udawania ich, nie ma też prostego sposobu do wywoływania kodu za ich pomocą. W niektórych przypadkach, przykładem może tu być czujnik oświetlenia, w który Twój laptop może być wyposażony, system Windows 8 z tego czujnika skorzysta. Jednak w innych sytuacjach, np. kiedy mowa jest o przyspieszeniomierzu lub żyroskopie, jedynym sposobem



Rysunek 11.18. Symulowanie danych lokalizacyjnych

Tabela 11.1. Czujniki w przestrzeni Windows.Devices.Sensors

Czujnik	Możliwość
Przyspieszeniomierz	Podaje wartości przyspieszenia w kierunkach x, y oraz z. Często używany do wykrycia potrząśnięcia urządzeniem w celu wykonania określonych czynności.
Kompas	Podaje informacje o kierunku na północ rzeczywistą lub północ magnetyczną, jeżeli jest obsługiwana przez czujnik.
Żyroskop	Podaje informacje o prędkości kątowej urządzenia.
Poziomnica	Podaje dane o kącie obrotu względem osi x, y i z (<i>pitch</i> , <i>roll</i> , <i>yaw</i>).
Czujnik oświetlenia	Podaje dane o natężeniu oświetlenia otoczenia, które mogą być użyte do ustawienia kontrastu i jasności wyświetlacza urządzenia.
Czujnik położenia	Zaawansowany czujnik używany najczęściej przez aplikacje wymagające dokładnych danych o kierunku (np. gry).
Prosty czujnik położenia	Podaje bieżące (ogólne) dane o kącie obrotu urządzenia oraz o położeniu przednią stroną w dół lub w górę.

jest użycie odpowiedniego urządzenia lub czujnika¹⁰. Jeżeli zamierzasz utworzyć aplikację wykorzystującą jedną lub kilka powyższych funkcjonalności, bardzo zalecam zainwestowanie w urządzenie, które dostarczy rzeczywiste informacje podczas pracy z nim. Bez konkretnych informacji o rzeczywistym czujniku narażasz się na duże ryzyko zbudowania wadliwej lub niekompletnej aplikacji.

Praca z czujnikiem oświetlenia

Pierwszym czujnikiem, któremu się przyjrzymy, jest czujnik oświetlenia, oferujący dane o natężeniu światła odbieranego przez urządzenie. Informację tę mogą dalej wykorzystać programiści do dostosowania kontrastu elementów swoich aplikacji lub wykonania innych czynności ułatwiających użytkownikom interakcję z urządzeniem w różnych okolicznościach, np. na leżaku na świeżym powietrzu podczas czytania tej książki z tabletem Windows 8 w jednej ręce i szklanką martini w drugiej¹¹. Zaczniemy od uzyskania referencji do bieżącego czujnika:

```
var lightSensor = Windows.Devices.Sensors.LightSensor.getDefault();
lightSensor.reportInterval = lightSensor.minimumReportInterval;
```

Podczas pracy z czujnikami metoda `getDefault()` wygląda znajomo, ponieważ za jej pomocą otrzymasz listę wszystkich czujników będących elementami przestrzeni `Windows.Devices.Sensors`. Inną wspólną cechą jest właściwość `reportInterval`, zawierająca wartość całkowitą oznaczającą liczbę milisekund pomiędzy kolejnymi danymi otrzymywanymi z czujnika. Wartością domyślną jest 16 ms, równa właściwości `minimumReportInterval`.

Kiedy mamy referencję do czujnika, następnym krokiem jest odczyt bieżących wskazań za pomocą funkcji `getCurrentReading()` lub dodanie nasłuchu zdarzenia `readingchanged` do obiektu `lightSensor`. Obie czynności są dostępne dla każdego czujnika w przestrzeni `Windows.Devices.Sensors`.

```
var reading = lightSensor.getCurrentReading(); // Pobranie pojedynczego odczytu
// Kod uruchamiany po każdej zmianie odczytu
lightSensor.addEventListener('readingchanged', function (event) {
    var lux = event.reading.illuminanceInLux.toFixed(2);
    document.getElementById('light').innerText = lux;
    adjustStylesheet(lux);
});
```

Za każdym razem, gdy obiekt `lightSensor` wykryje zmianę w natężeniu oświetlenia otoczenia, uruchomi się procedura nasłuchu zdarzenia. W tym miejscu odwołujemy się do obiektu `reading` należącego do argumentu `event`, aby odczytać właściwość `illuminanceInLux`, czyli podstawowe wskazania z obiektu `lightSensor`¹², i przekształcić ją w dwucyfrową liczbę. Następnie umieszczamy wartości zmiennej `lux` w elemencie naszej strony i wywołujemy lokalną funkcję dostosowującą stan aplikacji na podstawie odczytanych danych:

¹⁰ Słyszałem o możliwości przetestowania czujników w systemie Windows 8 za pomocą układów oferowanych przez zewnętrzne firmy, takie jak np. Digi-Key, opisanych pod adresem <http://digikey.com/scripts/DkSearch/dksus.dll?lang=en&keywords=STEVAL-MKI119V1&x=18&y=18&cur=USD> (<http://tinysells.com/213>) oraz za pomocą aktualizacji oprogramowania firmware dostępnego pod adresem <http://www.st.com/internet/evalboard/product/252756.jsp> (<http://tinysells.com/214>).

¹¹ Nie ja jeden zawsze czytam w ten sposób, prawda?

¹² Luks jest standardową jednostką oświetlenia. Służy do wyrażenia intensywności światła postrzeganej przez ludzkie oko. Jeżeli ten opis Ci nie wystarcza, możesz zajrzeć do Wikipedii pod adres [http://pl.wikipedia.org/wiki/Luks_\(fotometria\)](http://pl.wikipedia.org/wiki/Luks_(fotometria)) (<http://tinysells.com/215>).

```
function adjustStylesheet(lux) {
    var body = document.body;

    if (lux > 1000 || lux < 40) {
        WinJS.Utilities.addClass(body, 'light');
    } else {
        WinJS.Utilities.removeClass(body, 'light');
    }
}
```

Jeżeli wartość zmiennej lux jest większa od 1000 lub mniejsza od 40, oznacza to, że użytkownik znajduje się w bardzo jasnym lub bardzo ciemnym otoczeniu, dlatego dodajemy do ciała naszej strony klasę, która sprawi, że jej szczegóły będą lepiej widoczne:

```
.light {
    color: #000;
    background-color: #fff;
}
```

W zasadzie ustawiamy czarny kolor tekstu i biały kolor tła. Jest to prosty i skuteczny sposób dostosowania aplikacji do oświetlenia (patrz rysunek 11.19).



Rysunek 11.19. Użycie obiektu LightSensor do dostosowania wyglądu aplikacji

Na rysunku po lewej stronie przedstawiam naszą aplikację RSS Reader przy silnym lub słabym oświetleniu, natomiast po prawej mamy domyślny widok, jaki użytkownik widzi, gdy oświetlenie jest w średnim zakresie. Trzeba przyznać, że jest to prosty przykład i, dostosowując aplikację do oświetlenia, należy uwzględnić coś więcej niż tylko kolor tekstu i tła ciała aplikacji. Niemniej jednak ten przykład pokazuje, jak prosta jest praca z obiektem LightSensor w aplikacjach w stylu Windows 8.

Praca z przyspieszeniemierzem

Następny w kolejce jest przyspieszeniomierz, czyli czujnik wykrywający ruch oraz prędkość i dostarczający aplikacji dane o tych wielkościach. Jednym z najczęstszych zastosowań przyspieszeniomierza w dzisiejszych aplikacjach jest reakcja na potrząśnięcie urządzeniem przez użytkownika. Uwzględnimy ją, umieszczając w aplikacji RSS Reader funkcjonalność „potrząśnij i odśwież”. Scenariusz jest prosty. Kiedy użytkownik potrząśnie urządzeniem, odświeżymy bieżący kanał.

```
var accelerometer = Windows.Devices.Sensors.Accelerometer.getDefault();
```

Podobnie jak `lightSensor`, referencję `accelerometer` do obiektu `Accelerometer` można otrzymać z metody `getDefault`. Mając referencję do bieżącego czujnika, dodajemy nasłuch zdarzenia:

```
accelerometer.addEventListener('readingchanged', function (event) {
    var accelData = document.getElementById('accelData');

    accelData.innerHTML = "X: " + event.reading.accelerationX.toFixed(4) +
        " | Y: " + event.reading.accelerationY.toFixed(4) +
        " | Z: " + event.reading.accelerationZ.toFixed(4);
});
```

Czujnik `Accelerometer` podaje wartości przyspieszenia urządzenia w kierunkach X, Y oraz Z, które wykorzystamy do uaktualnienia tagu `<div>` i przedstawienia wyników użytkownikowi. Aby wykryć zdarzenie potrząśnięcia, możemy śledzić zmiany w wartościach `AccelerationX`, Y oraz Z dostarczanych przez czujnik i decydować, czy te zmiany przekraczają ustalony próg, wymagany do wywołania odświeżenia. Ponieważ jednak potrząśnięcie urządzeniem jest bardzo popularną formą korzystania z niego, dlatego zamiast powyższego sposobu można nasłuchiwać zdarzenia `shake`, oferowanego przez środowisko WinRT:

```
syn = new Windows.Web.Syndication.SyndicationClient();
url = new Windows.Foundation.Uri(this.feed.url);
accelerometer.addEventListener('shaken', function (e) {
    element.querySelector(".pagetitle").innerHTML = "POTRZĄŚNIĘCIE! Odświeżanie...";
    syn.retrieveFeedAsync(url)
        .then(processPosts, downloadError)
        .done(function () {
            element.querySelector(".pagetitle").innerHTML = options.feed.title;
        });
});

function processPosts(request) { ... }
```

Kiedy pojawi się zdarzenie `shake`, zmienimy na chwilę tytuł strony, aby poinformować użytkownika, że trwa odświeżanie, po czym wywołamy klienta syndykacji w celu pobrania zestawu aktualnych wpisów z kanału. Rezultat porządnego potrząśnięcia urządzeniem przedstawiam na rysunku 11.20.

Praca z kompasem

Innym czujnikiem, oferowanym programistom przez system Windows 8, jest kompas, dostarczający dane liczbowe oznaczające kierunek (lub odchylenie) w stopniach bieżącego położenia urządzenia względem magnetycznej lub rzeczywistej północy. Jeżeli czujnik jest zaawansowany, wyznacza północ magnetyczną. W przeciwnym przypadku wyznacza północ rzeczywistą¹³. Aby zobaczyć kompas w akcji, zbudujemy... kompas w stylu Windows 8. Będzie to surowy kompas z różą wiatrów narysowaną w grafice SVG, ale nadający się do naszego przykładu.

```
<h2>Wskazanie kompasu: <span id="compassData"></span></h2>
<div id="compass">
    <embed src="images/Simple_compass_rose.svg" title="Kompas"
        type="image/svg+xml" />
</div>
```

¹³ Wartości wskazań północy rzeczywistej i magnetycznej są zazwyczaj bardzo podobne, ale północ magnetyczna jest dokładniejsza, ponieważ brana jest pod uwagę bieżąca deklinacja od rzeczywistej północy, która jest różna w zależności od krzywizny naszej planety. Nie wszystkie czujniki wskazują północ magnetyczną, ponieważ ta funkcjonalność zazwyczaj wymaga intensywnych operacji i skomplikowanego kompasu.



Rysunek 11.20. Przechwycenie zdarzenia potrząśnięcia za pomocą przyspieszeniometera

Tak jak w innych czujnikach, pobierzemy instancję obiektu Compass za pomocą metody `getCurrent`, a następnie skonfigurujemy nasłuch odpowiadający na zmianę kierunku:

```
var compassDataLabel = document.getElementById('compassData');
var reading;
compass = Windows.Devices.Sensors.Compass.getDefault();

if (compass) {
    compass.reportInterval = 50;
    compass.addEventListener("readingchanged", function (event) {
        reading = event.reading;
        var heading = reading.headingMagneticNorth !== null ?
            reading.headingMagneticNorth.toFixed(2) :
            reading.headingTrueNorth.toFixed(2);
        compassDataLabel.innerText = heading;
        rotateCompass(heading);
    });

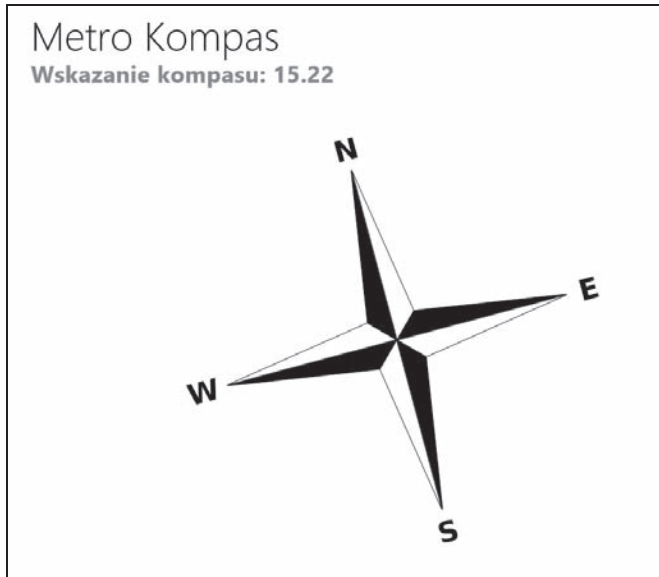
    reading = compass.getCurrentReading();
    compassDataLabel.innerText = reading.headingMagneticNorth ||
        reading.headingTrueNorth;
}
```

Jeśli nastąpi zmiana w bieżącym położeniu użytkownika, możemy użyć obiektu `reading` w obiekcie `event` do określenia bieżącego kierunku w stopniach — odczytując wartość właściwości `headingMagneticNorth` lub `headingTrueNorth` — i skorzystać z tych danych do obrócenia kompasu:

```
function rotateCompass(heading) {
    var compass = document.querySelector('embed');

    compass.setAttribute('style',
        "transform: rotate(-" + Math.floor(heading) + "deg);");
}
```

Tutaj stosujemy wartość zmiennej `heading` do wykonania operacji obrócenia róży wiatrów narysowanej w grafice SVG, która — z niewielką pomocą magicznych sztuczek z transformacją stylów CSS — wywoła efekt działającego kompasu, przedstawionego na rysunku 11.21.



Rysunek 11.21. Użycie kompasu

Nie jest to kompas, który umieściłbym w sklepie i sprzedał jutro za kilka groszy, ale ilustruje zasadę. Poza tym, nie chcę odkrywać sekretów mojej aplikacji, więc dlaczego miałbym to teraz robić?

Praca z prostym czujnikiem położenia

Ze względu na szerokie zastosowanie danych o położeniu urządzenia środowisko WinRT oferuje w przestrzeni `Sensors` dwa czujniki położenia: `orientationSensor` oraz `SimpleOrientationSensor`. Czujnik `orientationSensor` powinien być stosowany w grach i aplikacjach wymagających szczegółowych danych o położeniu urządzenia, natomiast czujnik `SimpleOrientationSensor` jest przewidziany do określania stanu położenia (obrócone o 90 stopni, przodem w dół itp.).

```
var sensors = Windows.Devices.Sensors;
var status = document.querySelector('#status');
orientationSensor = sensors.SimpleOrientationSensor.getDefault();

orientationSensor.addEventListener("orientationchanged", function(event) {
    status.innerText = "Przodem w dół? " +
        e.orientation === sensors.SimpleOrientation.facedown;
});
```

W odróżnieniu od danych liczbowych dostarczanych przez inne czujniki, `SimpleOrientationSensor` podaje tylko jedną z sześciu wartości enumeracji, oznaczających położenie:

- `notRotated` (nieobrócony),
- `rotated90DegreesCounterclockwise` (obrócony o 90 stopni przeciwnie do ruchu zegara),
- `rotated180DegreesCounterclockwise` (obrócony o 180 stopni przeciwnie do ruchu zegara),
- `rotated270DegreesCounterclockwise` (obrócony o 270 stopni przeciwnie do ruchu zegara),
- `faceup` (przodem do góry),
- `facedown` (przodem w dół).

Oprócz użycia tego czujnika do reakcji na zmianę położenia urządzenia, co możesz osiągnąć za pomocą procedury `updateLayout` kontrolki `Page` środowiska `WinJS` lub zapytania o media w stylach CSS, wartości `faceup` oraz `facedown` oferują Twojej aplikacji kilka ciekawych możliwości. Możesz np. wykrywać położenie i automatycznie wstrzymywać grę użytkownika lub zawieszać jakieś funkcjonalności aplikacji intensywnie wykorzystujące baterię. Albo jeszcze lepiej, możesz użyć zdarzenia `facedown` do wyczyszczenia ekranu użytkownika w aplikacji rysowanki `Etch-a-Sketch`.

Praca z innymi czujnikami

Oprócz czujników opisanych w tym rozdziale, środowisko `WinRT` oferuje dostęp do poziomnicy, żyroskopu i czujników położenia, i każdy z nich może być użyty do wzbogacenia interakcji z Twoją aplikacją. Podobnie jak w przypadku czujników opisanych wcześniej, użycie każdego z nich jest bardzo proste i wymaga pobrania referencji (za pomocą metody `getX()`), zdefiniowania jednego lub kilku nasłuchów zdarzeń oferowanych przez czujnik i wreszcie użycia danych przez nie dostarczanych do zmiany stanu lub działania Twojej aplikacji. Praca z kilkoma bardziej zaawansowanymi czujnikami może być zniechęcająca, gdy otrzymasz surowe dane wymagające wielu obliczeń, ale obsługa ich da Ci możliwość implementacji wielu ciekawych i zaskakujących interakcji w Twojej aplikacji w stylu `Windows 8`.

Gdzie jesteśmy?

W całym rozdziale była mowa o urządzeniach w systemie `Windows 8`, o tym, jak użytkownicy korzystają z nich oraz jak urządzenia komunikują się ze światem. Najpierw dowiedziałeś się, jak system `Windows 8` stawia na równi dotyk z interakcjami za pomocą myszy i klawiatury oraz jak ambitnie platforma obsługuje równorzędnie wszystkie trzy metody. Następnie poznałeś standardowe sposoby obsługi dotyku w aplikacjach w stylu `Windows 8`, jak również różne kontrolki przyjazne w dotyku, oferowane przez platformę. Wiesz już, że aplikacje powinny wciąż umożliwiać obsługę myszy i klawiatury użytkownikom nieposiadającym urządzeń dotykowych.

Zapoznałeś się też z możliwościami urządzeń i czujników, poczynawszy od popularnych funkcjonalności, takich jak nagrywanie dźwięku, poprzez drukowanie i zastosowanie usług lokalizacyjnych. Rozdział został podsumowany dyskusją o czujnikach, wraz z opisem wspólnego szablonu interakcji oferowanych przez elementy przestrzeni `Windows.Devices.Sensors` i kilkoma przykładami pokazującymi czujniki w akcji.

Teraz, kiedy znasz już wszystko, co jest Ci potrzebne do wykorzystania potęgi dotyku, możliwości urządzenia i czujników, możesz przejść do następnego rozdziału, w którym opisuję, jak rozszerzyć możliwości platformy `Windows 8` poprzez integrację w Twojej aplikacji rozszerzeń w języku `C++`.



Skorowidz

A

ABI, Application Binary Interface, 330
adaptacja do rozdzielczości, 90
adres pliku wideo, 142
AJAX, 48, 227
aktualizacja
 interfejsu użytkownika, 62
 kafelka, 276, 279
 treści, 283
aktywacja
 aplikacji, 296
 plików, 216
 WinJS, 406
aktywny roaming, 225
analiza wyników
 JSON, 229
 XML, 228
animacja, 169, 175, 183
 CSS, 186
 elementów, 189
 strony, 190
 SVG, 173
 WinJS, 388
aplikacja
 dotykowa, 300
 ekranu blokowania, 291
 nawigacji, 31, 32
 podziału, 31, 51
 pusta, 31
 RSS Reader, 246, 269, 272
 siatkowa, 31, 120
 SkyDrive, 219
 stałego podziału, 31

aplikacje
 arkusze stylów, 445
 certyfikacja, 380
 CSS, 444
 dźwięk alarmu, 394
 eksponowanie, 399
 funkcjonalności zaawansowane, 375
 klasyfikacja wiekowa, 375
 komercyjne, 398
 licencja na użytkowanie, 389
 natywne, 427
 odbior zapłaty, 400
 odrzuć, 381
 oferty, 393
 okres próbny, 389
 opis, 377
 podpisy cyfrowe, 375
 przesłanie, 365
 przesyłanie, 373, 397
 przesyłanie aktualizacji, 382
 reklamy, 382
 rezerwacja nazwy, 366
 Sklep Windows, 19, 28, 389
 sprzedaż, 374, 398
 strona opisu, 379
 struktura, 427
 style CSS, 434
 śledzenie, 398
 testy lokalne, 368
 tworzenie pakietu, 370
 wykonywanie zrzutów, 378
 zakup, 395
 załadowanie pakietu, 376
 zarządzanie, 398

architektura wspólnego języka, CLI, 330
argumenty funkcji, 418
asynchroniczność, 357
Atom, 232
atrybut
 data-win-bind, 59, 64
 data-win-control, 34, 76
 data-win-options, 34
 msAudioDeviceType, 151
 msZoom, 150
atrybuty
 danych, 431
 znacznika <video>, 141
audio, 139

B

biblioteka, 214
 równoległych wzorców, PPL, 359
 System.Graphics.Imaging, 183
 Windows.Graphics.Imaging, 180, 190
 WinJS, 33
 WinJS Animation, 187–190
 WinRT, 194
biblioteki
 mediów, 154
 systemowe, 161
 uruchomieniowe Windows, 330
bindowanie, 57, 72
 danych, 41
 dwustronne, 58
 jednokierunkowe, 58

bindowanie
 jednostronne, 58
 kolekcji, 66
 obiektów, 58, 63

Blend, 38
 bindowanie danych, 41
 dodawanie kontrolki, 40
 dodawanie reguły, 92
 edytowanie zawartości
 szablonu, 42
 kontrolowanie orientacji, 96
 linie siatki, 101
 przeglądanie zawartości
 szablonu, 42
 style, 108
 style Flexboks, 106
 tworzenie szablonu, 41
 wybieranie reguły CSS, 23
 zapytania medialne, 92
 zarządzanie stylami, 43
 zmiana stanu widoku, 98

błędy, 235, 371
 pobierania, 228
 wykonania, 361

boczny pasek menu, 303

brzegi ekranu, 303

BUM, Badge Update Manager, 286

C

CameraCaptureUI, 163

Canvas, 169, 174, 178, 182

certyfikacja
 aplikacji, 369
 klasyfikacji, 375

ciągi znaków, 347

CLI, Common Language
 Infrastructure, 330

CLR, Common Language Runtime,
 330

CMS, 444

COM, Component Object Model,
 330

CSS, Cascading Style Sheets, 42,
 430, 432
 animacje, 186
 dokładność, 443
 GRID, 54
 ostatnia reguła, 442
 przejścia, 185
 transformacje, 184
 ważność, 443

CSS3, 99

cykl życia aplikacji, 201, 203

czas życia zmiennych, 353

czcionka
 Calibri, 117, 125
 Cambria, 117
 Segoe UI, 116
 Segoe UI Symbol, 302

czcionki
 darmowe, 120
 ikonowe, 133, 137
 niestandardowe, 135
 web, 119
 z dopasowanymi regułami, 122
 z niedopasowanymi regułami,
 121

czujnik
 kompas, 322, 325
 oświetlenia, 322
 położenia, 322, 327
 poziomnica, 322
 przyspieszeniometer, 322, 324
 żyroskop, 322

czytnik RSS, 35, 47, 107, 195

D

dane lokalizacyjne, 316, 319

debugowanie, 36, 207, 421
 dostawcy kontaktów, 272
 sesji, 206

definiowanie
 ofert, 397
 stylów CSS, 441

deklaracja, 433

deklaracja @font-face, 119

deklaracje
 C++, 334
 WinRT, 334

delegaty WinRT, 351, 354

DLNA, 165

dodawanie
 efektów wizualnych, 148
 elementów, 45
 możliwości drukowania, 313
 obrazu reklamy, 386
 odwołania do pakietu SDK, 319
 projektu, 332
 reguły, 92

DOM, Document Object Model, 35

domknięcia, 351, 420

dostawca, 245

dostawca kontaktów, 267, 272

dostęp do
 bibliotek, 161, 216
 biblioteki obrazów
 użytkownika, 182
 internetu, 224
 kamery internetowej, 164
 pól, 335
 urządzeń Play To, 166

dostosowywanie
 typografii, 122
 wyglądu aplikacji, 324

dotyk, 299
 kontrolki HTML, 300
 kontrolki WinJS, 302

DPI, 90, 93

DRM, Digital Rights Management,
 375

drukowanie, 313, 314

drzewo DOM, 35

duplikacja zadań, 293

dynamiczne kafelki, 275

dziedziczenie prototypów, 415

dźwięk, 151

E

edytor manifestu, 33

efekt
 wypełnienia, 150
 zbliżenia, 149

efekty wizualne, 148

ekran
 blokowania, 291
 dotykowy, 303
 powitalny, 28
 startowy, 30, 244, 307

element iframe, 237, *Patrz także*
 znacznik

elementy
 HTML, 73
 HTML5, 73
 reguły text-shadow, 124

elipsa SVG, 171

enumeracja
 członków, 335
 PeriodicUpdateRecurrence, 282
 PositionStatus, 317

enumerátor
 AppBarIcon, 130, 132
 WinJS.UI.AppBarIcon, 133

F

film, 140
 dodawanie napisów, 145
 sterowanie, 144
 filtr, 221
 filtrowanie, 68, 69
 Flexboks, 103
 przeglądanie zdjęć, 104
 zmiana orientacji kontenera, 105
 format
 appx, 368
 Atom, 50
 JSON, 227, 425
 WebVTT, 145
 WinMD, 330
 WinRT, 330
 XML, 227
 fotogaleria, 110
 funkcja
 animation-duration, 187
 appIconStreamReference, 252
 bind, 420
 call, 419
 captureCamera, 163
 completed, 229
 copyAsync, 181
 costType, 225
 createFileAsync, 212
 createFiltered, 67
 createSorted, 67
 enterPage, 190
 exists, 214
 feedInvoked, 46
 getBitmapAsync, 264
 getCostType, 226
 getCurrentReading(), 323
 getImageData, 180
 getPixelDataAsync, 182
 getSettings, 199
 hasKey, 193
 importScripts, 291
 invertImage, 179
 manipulateCanvasCircle, 176
 matrix, 185
 pickImage, 179
 pickSingleFileAsync, 218
 postMessage, 241, 242
 processAll, 60
 querySelectorAll, 172
 ready, 39, 270
 renderAlarmData, 397

rotate, 185
 scale, 185
 setInterval, 172
 setSettings, 199
 skew, 185
 stringify, 192
 SyndicationClient, 50
 task<T>.get, 362
 translate, 185
 updateBadge, 287

funkcje
 asynchroniczne, 49
 lambda C++11, 352
 nasłuchujące, 357
 pomocnicze dla plików, 213
 pomocnicze dla sesji, 209
 funkcjonalności zaawansowane, 375
 funkcjonalność
 BUM, 286
 geolokalizacji, 316

G

geolokalizacja, 316
 głębokie odnośniki, 283
 gradient, 174
 grafika wektorowa, 169
 grupowanie, 69, 72
 grupowanie kontrolek, 308

H

historia nawigacji, 205, 206
 hosting elementów iframe, 237
 HTML, Hypertext Markup
 Language, 428
 HTML5, 139, 429

I

identyfikator
 contenthost, 34
 id, 406
 imgContainer, 103
 kafelka, 284
 this, 343, 354
 IDL, Interface Definition Language,
 358
 ikonografia, 126
 ikony enumeratora, 131
 Imaging API, 180

implementacja
 funkcjonalności oferty, 393
 licznika, 355
 wyszukiwania, 246
 informacje
 o lokalizacji, 320
 o zdarzeniu, 356
 inicjalizatory, 64
 instalacja poboczna, 368
 integracja z powłoką, 275
 interakcja z urządzeniem, 299
 interfejs
 ABI, 343
 API Geolocation, 316
 IAsyncAction, 358
 IAsyncInfo, 358
 IAsyncOperation, 358
 IAsyncOperation<T>, 358
 użytkownika aplikacji, 291
 interfejsy adaptacyjne, 103, 105
 internet, 223
 inwersja pikseli, 178

J

JavaScript, 334, 403–425
 daty, 410
 domknięcia, 420
 funkcje, 417
 klasy, 412
 moduły, 422
 obiekty, 409
 operatory, 408
 przestrzeń nazw, 423
 rodzaje zasięgu, 421
 serializacja, 425
 symulowanie działania klas, 415
 tablice, 411
 typy, 407
 wartości, 407
 wynoszenie, 422
 wyrażenia regularne, 410
 jednostka fr, 101
 język
 C++, 331
 definicji interfejsu, IDL, 358
 JavaScript, 334, 403–425
 metro, 18
 TTML, 145
 JSON, JavaScript Object Notation,
 425

K

kafelki, 194
 aktualizacje, 276, 279
 aktualizacje zaplanowane, 282
 dodatkowe, 283
 dynamiczne, 275
 obrazy, 280
 odwracanie, 281
 rozmiar, 276, 277
 znaczki, 286
 kanały RSS, 232
 kaskadowe arkusze stylów, CSS, 433
 kaskadowość reguł CSS, 442
 katalog Biblioteki, 214
 katalogi
 lokalne, 213, 216
 roamingowe, 216
 tymczasowe, 216
 klasa
 .win-star, 302
 ClockControl, 79, 82
 DataPackageView, 263
 StringReference, 348
 win-backbutton, 127
 WorkerGlobalScope, 290
 klasy WinRT
 metody, 336
 właściwości, 340
 klasyfikacja wiekowa, 375
 klatka kluczowa, 186
 klawiatura, 310
 kod semantyczny, 427
 kodowanie dźwięku, 312
 kody
 COM HRESULT, 338
 HRESULT, 339
 kolekcja arguments, 419
 kolekcje adaptacyjne, 108
 komentarze TODO, 251
 kompas, 322, 325
 kompilator C++11, 334
 komunikat MessageDialog, 317
 konfiguracja Neutral, 369
 konstruktor WinRT, 341
 konstruktory, 335, 413
 kontekst WWW, 239
 konto dewelopera, 366
 kontrakt, 224
 kontaktów, 265
 powłoki, 243, 246
 udostępniania, 252

Udział docelowy, 257
 wyszukiwania, 246
 kontrolka
 Bing Map, 239
 FileSavePicker, 161
 ListView, 40, 108, 306
 aktualizowanie, 112
 fotogaleria, 110
 wygląd domyślny, 111
 PageControlNavigator, 34
 SemanticZoom, 305, 308, 309
 SettingsFlyout, 195
 ToggleSwitch, 197
 zegara, 78
 kontrolki, 73
 HTML, 300, 301
 metody, 79
 niestandardowe, 78
 rozmiar, 302
 WinJS, 73, 75, 77, 302
 WinRT, 73, 74
 właściwości, 79
 wyboru napisów, 147
 wyboru plików, 154
 zdarzenia, 81
 kopiowanie, 182
 kopiowanie przy zapisie, 414
 koszt połączenia, 226
 kryptografia, 375
 krzywa Béziera trzeciego stopnia, 186

L

lambda, 351
 licznik, 354
 ligatura, 125
 lista
 bindowania, 65
 deklaracji, 247
 kanałów, 208
 kontaktów, 269
 kontraktów, 246
 sugestii, 251
 logo dla sklepu, 28
 lustrzane odbicie, 182

M

manifest, 28, 29
 manipulacja pikselami, 178, 180

mapowanie typów
 C++/CX, 344, 345
 WinRT, 344
 mapy Bing Maps, 319
 mechanizm
 tombstoning, 206
 view state, 86
 media, 139
 menadżer
 odwołań, 333
 BUM, 287
 metadane WinRT, 343
 metoda, 335
 as, 61
 createUrl, 156
 DataPackage, 253
 DataPackageView, 253
 getFilesAsync, 216
 getForCurrentView, 167, 314
 getInternetConnectionProfile, 224
 getPositionAsync, 317
 GetResults, 362
 processPosts, 49
 requestProductPurchaseAsync, 395
 slice, 419
 SyndicationClient, 51
 toStaticHTML, 49
 WinJS.Class.define, 80
 xhr, 230
 metody
 asynchroniczne, 358
 kontrolki, 79
 statyczne, 416
 synchroniczne, 358
 model widoku, 62
 moduły, 422
 modyfikator
 &, 342
 *, 342
 ^, 342
 const, 349
 możliwości
 sieciowe, 223
 urządzenia, 310
 MVVM, Model-View-ViewModel, 62
 mysza, 310

N

nadpisywanie stylów, 446
 nagłówki HTTP, 227
 nagrywanie filmów, 165
 napisy do filmów, 145
 narzędzia HTML, 38
 narzędzie
 MakeAppx.exe, 29
 SignTool.exe, 29
 WACK, 370
 Wyszukiwanie, 246
 nasłuch zdarzenia, 323
 nasłuch zdarzenia positionchanged, 318
 natywne rozszerzenia kodu, 329
 nawigacja, 35, 43, 254
 nazwa aplikacji, 366, 373
 numer
 ECCN, 375
 GUID, 212

O

obiekt
 Accelerometer, 325
 audioFile, 313
 BackgroundDownloader, 233
 captureSettings, 312
 Compass, 326
 Contact, 271
 CurrentAppSimulator, 389
 DataPackage, 255
 File, 181
 FileOpenPicker, 155
 Geolocator, 316, 318
 JSON, 192
 KnownFolders, 216
 licenseInfo, 392
 LicenseInformation, 395
 lightSensor, 323
 LightSensor, 324
 location, 317
 MediaCapture, 312
 MediaCaptureInitialization
 ↳ Settings, 311
 MediaControl, 153
 Navigation, 205
 Promise, 234
 responseXML, 235
 ScheduledTileNotification, 282
 SettingsPane, 194

StorageFile, 155, 218
 SyndicationClient, 231
 ToastNotificationManager, 296
 URL, 156
 WebUIBackgroundTaskInstance, 290
 WorkerGlobalScope, 290
 XHR, 228, 230, 231
 XMLHttpRequest, 48, 227
 obiekty
 Array, 349
 funkcyjne, 351
 pierwszego poziomu, 213
 WinRT, 341
 obrazy kafelka, 280
 obserwacja zmian położenia, 318
 obsługa
 aplikacji za pomocą zadań, 288
 dotyku, 299, 306
 formatów danych, 262, 263
 komunikatów, 242
 kontraktu wyboru kontaktów, 268
 lokalizacji geograficznej, 315
 mechanizmu view state, 86
 nawigacji, 35
 odrzućenia aplikacji, 381
 rozdzielczości, 86
 sieci, 48
 trybu przypięcia, 56
 wartości lambda, 352
 zdarzeń, 355, 357
 odczyt danych, 213
 odnośnik QuickLink, 261
 odtwarzanie
 dźwięku, 152
 filmu, 140
 odwołanie do
 pakietu, 385
 projektu C++, 331
 odwracanie kafelka, 281
 oferty w aplikacji, 393
 okna wybierania plików, 217
 okno
 dialogowe WinRT, 74
 Menadżer odwołań, 333
 wyboru katalogu, 221
 wyboru pliku, 156, 218
 wyboru pliku do zapisu, 219
 zapisu pliku, 162
 opcje debugowania, 36
 operacja udostępnienia, 261

operacje

 asynchroniczne, 358, 362
 odczyt, 218
 otwieranie, 218
 sieciowe, 223
 zapis, 218
 operator
 [], 349
 ===, 343
 kropka, 335
 ref new, 343
 opis
 aplikacji, 377
 testów, 370
 orientacja, 95
 osadzanie kontrolki Bing Map, 239
 oświetlenie, 322

P

pakiet, 376
 Bing Maps SDK, 319
 SKD, 290
 Windows Advertising SDK, 384
 panel
 ustawień, 193–195
 wyszukiwania, 247
 pasek
 AppBar, 304
 Uprawnienia, 318
 Play To, 165
 plik
 backgroundtasks.js, 288
 data.js, 53, 199
 default.html, 33
 home.css, 42
 home.js, 35
 manifestu, 28
 package.appxmanifest, 246
 package.appxmanifest, 161, 182, 214, 255, 267
 searchResults.js, 251
 shareTarget.css, 257
 shareTarget.html, 257
 shareTarget.js, 257–260
 ui.js, 83
 WindowsStoreProxy.xml, 392
 pliki, 211
 .appx, 29
 .appxmanifest, 32
 .appxupload, 376

pliki
 .foo, 214, 216
 .jsproj, 31
 .svg, 177
 .zip, 368
 WinMD, 330
 płatności, 401
 podgląd wydruku, 314
 polecenia
 na pasku, 305
 Resume, 206
 web, 227
 polecenie
 Add-AppxPackage, 29
 Get-AppxPackage, 29
 Suspend and shutdown, 206
 położenie, 322, 328
 postęp
 pobierania, 228
 udostępniania, 265
 powiadomienia, 296
 ekranu blokady, 292
 w pasku przewijania, 294
 zaplanowane, 297
 powłoka Windows 8, 243
 poziomnica, 322
 PPL, Parallel Patterns Library, 359
 prefiks ms-appx-web, 240
 program Blend, 38
 projekt
 adaptacyjny, 103
 C++, 331
 JavaScript, 332
 responsywny, 103
 projektowanie
 komercyjnych aplikacji, 398
 WYSIWYG, 38
 proporcja obrazu, 149
 protokół
 Atom, 232
 Odata, 232
 RSS, 233
 prototypy, 413
 prototypy obiektów, 412
 przechwytywanie
 mediów, 162
 wyjątków, 338
 zmiennych, 353
 przeciążanie
 konstruktorów, 337
 metod, 337
 w JavaScript, 337

przeglądanie zdjęć, 104
 przejścia CSS, 185, 186
 przejście z rotacją, 186
 przestrzenie nazw WinJS, 423
 przestrzeń nazw
 Data, 198
 Microsoft.Maps, 241
 RssReader, 34
 Windows.Data.Xml.Dom, 235
 Windows.Devices.Sensors, 321
 Windows.Foundation, 50
 Windows.Web.Syndication, 50
 Windows.Storage, 216
 Windows.Storage.Application
 ↳ Data, 192
 Windows.UI.WebUI, 201
 Windows.Web.Syndication,
 231
 WinJS.Binding, 60
 przesyłanie
 aktualizacji, 382
 aplikacji, 365, 367, 373, 397
 mediów, 165, 167
 przyspieszoniemierz, 322, 324
 pseudoelementy, 128
 pseudoklasy, 128
 pulpit Centrum deweloperów, 373

R

raportowanie postępu
 udostępniania, 265
 rednerowanie grafiki, 175
 reguła
 !important, 443
 @font-face, 119
 resolution, 95
 rejestracja dźwięku, 312
 reklamy, 382, 383
 obraz, 386
 tekst, 387
 responsywność interfejsu
 użytkownika, 183
 rezerwacja nazwy, 367
 robienie zdjęć, 164
 rodzaje
 czcionki Segoe UI, 116
 powiadomień, 296
 rozdzielczość wysokiej jakości, 94
 rozmiary ekranów, 88
 rozszerzenie C++/CX, 330, 334

równość
 dokładna, 409
 z konwersją typów, 409
 RSS, Really Simple Syndication, 31
 rysowanie, 169

S

SEH, Structured Exceptions
 Handling, 338
 selektor, 433
 kontaktów, 267
 kontraktów, 265
 selektory
 CSS, 435
 CSS zaawansowane, 437
 CSS3, 439
 serializacja, 425
 JSON, 241
 stanu aplikacji, 210
 sesja, 203
 sesja w formacie JSON, 210
 siatka, 100
 sieci prywatne, 223
 sieć
 LAN, 224
 mobilna, 224
 skojarzenia typów plików, 215
 skróty klawiaturowe
 Alt+Tab, 236
 Ctrl+Enter, 310
 Win+H, 252
 Win+kropka, 236
 Win+Tab, 236
 Win+Z, 310
 SkyDrive, 219
 słowo kluczowe
 class, 336
 mutable, 353
 nullptr, 342
 public, 336
 ref, 336
 sealed, 336
 struct, 350
 var, 407
 sortowanie, 67, 69
 specyfikacja
 CSS3, 100
 układu siatkowego, 100
 sprawdzanie
 aktywacji powiadomienia, 297
 poprawności pliku appx, 30

sprzedaż, 374, 398
standard RSS, 232

stany
 aplikacji, 191, 201
 certyfikacji aplikacji, 380
 widoku, 97

sterowanie
 dźwiękiem, 154
 filmem, 144

strict, 423

strona
 itemsPage.htm, 55
 itemsPage.html, 54
 kontaktów, 269
 nawigacji, 195
 opisu aplikacji, 377
 wiadomości RSS, 106

struktura aplikacji, 427

struktury WinRT, 350

styl metro, 18

style CSS, 42, 73, 432

stylowanie, 38

sugestie wyszukiwania, 251

SVG, Scalable Vector Graphics, 169

symulator, 37

symulator układu, 38

symulowanie

 danych, 390
 danych lokalizacyjnych, 322
 działania klas, 415
 zakupu, 396

syndykacja, 231

system zarządzania treścią, CMS, 444

szablon, 70

 Aplikacja podziału, 51
 Aplikacja siatkowa, 120
 Formant, 196
 kontraktu wyszukiwania, 249

szablony

 aplikacji, 31
 dla projektów, 38

Ś

ścieżka localFolder, 212

śledzenie

 aktualnie wybranego elementu,
 253
 aplikacji, 398

środowisko uruchomieniowe

 WinRT, 19
 wspólnego języka, CLR, 330

środowisko Web Workers, 290

T

tabele, 349

Tablica znaków, 128, 129, 135

tablice, 411

technologia AJAX, 48

testowanie, 347

 aplikacji, 368

 czujników, 323

 okresu próbnego, 391

testy WACK, 370

transfer danych w tle, 232

transformacje CSS, 184

treść

 HTML, 237, 239

 WWW, 236

tryb

 debugowania, 37
 natychmiastowy, 173
 portretu, 111
 przypięcia, 56
 standardów, 423
 utrzymania, 170

TTML, 145

TUM, 287

tworzenie

 adaptacyjnych kolekcji, 108
 animacji CSS, 188
 delegatów WinRT, 354
 dźwięku, 151
 interfejsów adaptacyjnych, 103
 konta programisty, 366
 niestandardowych kontroltek,
 142
 panelu ustawień, 196
 projektu, 31
 przyjaznych w dotyku aplikacji,
 303
 przyjaznych w dotyku
 interakcji, 305
 szablonu, 41
 zadania w tle, 290

typ

 Date, 410
 Object, 412

typografia, 115

typy

 JavaScript, 407
 treści dla reklam, 384
 WinRT, 343

U

uchwyty, 342

udostępniane typy danych, 253

udostępnianie, 261, 265

 elementu, 256

 zdjęcia, 245

 artykułu, 260

udział docelowy, 245, 255

układ, 85

 Flexbox, 104

 siatkowy, 100

 w Windows 8, 86

 wielokolumnowy, 105

układy adaptacyjne, 103

urządzenia

 deklarowanie możliwości, 311
 rejestrujące, 311

usługa, 375

 Live Connect Services, 375

 lokalizacyjna systemu, 317

 nawigacji, 254

 przyszłego dostępu, 221

 syndykacji, 232

ustawianie stylów CSS, 434

ustawienia, 191

 aplikacji, 197

 lokalne, 193

 roamingowe, 193, 201

 uprawnień, 318

uwierzytelnianie, 223

użycie kompasu, 327

V

Visual Studio 2012, 31

W

W3C, World Wide Web
 Consortium, 75

WACK, Windows App
 Certification Kit, 370

wartości, 433

wartości tekstowe, 347

warunki, 289

WebVTT, 146

wiadomości RSS, 31

wideo, 139

widok

 ekranu startowego, 307
 przypięty, 100

WinJS, 33, 416
 animacje elementów, 189
 animacje strony, 190
 funkcje pomocnicze, 209, 213
 stany, 204
 zdarzenia cyklu życia, 203, 204
 WinMD, Windows Metadata, 330
 WinRT, Windows Runtime, 19, 33, 50
 asynchroniczność, 357
 delegaty, 351, 354
 klasy, 335
 konstruktor, 341
 obiekty, 341
 struktury, 350, 351
 tabele, 349
 typy, 343
 uchwyt, 342
 współbieżność, 357
 zdarzenia, 355
 właściwości, 335, 433
 kafelka aplikacji, 276
 kontrolki, 79
 OpenType, 126
 układu wielokolumnowego, 109
 właściwość
 animation-name, 187
 args.detail.files, 217
 BackgroundCapableMedia, 151
 CameraCaptureUIMode, 163
 CSS transform, 184
 fileTypeFilter, 155, 221
 icon, 130
 isTrial, 391
 letter-spacing, 123
 msAudioCategory, 151
 msHorizontalMirror, 148
 msZoom, 149
 path, 212
 suggestedStartLocation, 155, 158
 text-shadow, 124
 word-spacing, 122
 włączanie kanału, 199
 WOFF, Web Open Font Format, 120
 WRL, Windows Runtime Library, 330
 wskaźnik postępu pobierania, 234
 współbieżność, 357, 362

wybieranie
 katalogu, 220
 kontaktów, 268
 plików, 154, 158, 218, 220
 typ FileSavePicker, 160
 typ FolderPicker, 160
 trybu debugowania, 37
 wydajność aplikacji, 177
 wygląd
 mediów, 142
 wygląd strony, 432
 wyjątek, 338, 362
 wyjątki WinRT, 338
 wyłapywanie zmiennych, 420
 wyniki
 testów WACK, 371
 wyszukiwania, 248
 wynoszenie, 422
 wyrażenia regularne, 410
 wyszukiwanie, 246, 248
 wyświetlanie
 powiadomień, 295
 znaczka, 287
 wywołanie zdarzenia kontrolki, 47
 wyzwalacz, 289
 wyzwalanie zadań, 288

Z

zadania w tle, 288, 290
 zakup aplikacji, 395
 zapytanie
 lokalizacyjne, 317
 o plik, 217
 medialne, 92, 94
 medialne CSS, 90
 zarządzanie
 prawami cyfrowymi, DRM, 375
 stylami CSS, 42
 zasięg
 funkcji, 421
 globalny, 421
 zawartość kanału RSS, 50, 51
 zdarzenia, 335
 asynchroniczne, 207
 cyklu życia WinJS, 204
 kontrolki, 81
 systemowe, 288

WebUIApplication, 201
 WinRT, 355, 357
 zdarzenie
 activated, 217
 aktywacji, 202
 click, 155
 datarequested, 254
 networkstatuschanged, 224
 positionchanged, 318
 potrząśnięcia, 326
 readingchanged, 323
 ready, 155
 zdjęcia, 164
 zestaw
 certyfikacji aplikacji, 369
 stylistyczny, 126
 zmiana
 położenia, 318
 rozdzielczości, 89, 91
 stanu widoku, 98
 układu, 110
 zmienna this, 290
 zmienne JavaScript, 407
 znaczek z symbolem, 286
 znacznik
 <blockquote>, 123
 <body>, 101
 <canvas>, 174, 180
 <div>, 305
 , 115, 139, 189
 <link>, 445
 <svg>, 169
 <track>, 146
 <video>, 140–143
 znaczniki
 dla mediów, 431
 semantyczne, 429
 znak *, 221
 zrzuty aplikacji, 378

Ż

żądanie
 aktualizacji, 282
 pozycji, 317
 żyroskop, 322

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**

Wykorzystaj potencjał JavaScript w Windows 8!

Czy pamiętasz czasy, kiedy użytkownicy blokowali JavaScript w przeglądarkach? Być może trudno w to uwierzyć, ale jeszcze całkiem niedawno nikt nie wierzył, że język ten osiągnie jakikolwiek sukces na rynku. Współczesne atrakcyjne, interaktywne i pełne możliwości strony WWW nie mogłyby bez tego języka istnieć. Znajduje on zastosowanie również w wielu innych miejscach — czasami wręcz zaskakujących. Dowiedz się, jak wykorzystać go do tworzenia aplikacji dla Windows 8!

W trakcie lektury tej książki nauczysz się używać kontrolek, tworzyć zaawansowane układy oraz korzystać z materiałów multimedialnych. Ponadto dowiesz się, jak komunikować się z siecią, integrować z powłoką oraz wykorzystywać interfejs dotykowy i natywny kod. Na koniec zobaczysz, jak poświęcony programowaniu czas sprawnie przekuć na sukces finansowy. Przygotujesz aplikację do publikacji i udostępnisz ją w Sklepie Windows oraz poznasz zasady publikowania reklam. Ta książka jest niezastąpionym źródłem informacji dla wszystkich programistów chcących tworzyć pomocne aplikacje w języku JavaScript. Twój sukces jest w Twoich rękach!

Dzięki tej książce:

- poznasz podstawy języka JavaScript i sposób wykorzystania go w systemie Windows 8
- zbudujesz intuicyjny interfejs użytkownika
- uzyskasz dane geolokalizacyjne z systemu
- opublikujesz Twoją aplikację w Sklepie Windows

 **Addison-Wesley**
Pearson Education

helion.pl
księgarnia
internetowa

Nr katalogowy: 17375



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/novosci>

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

Informatyka w najlepszym wydaniu

sięgnij po **WIĘCEJ**



KOD KORZYŚCI

cena 77,00 zł

ISBN 978-83-246-7564-7



9 788324 675647

PEARSON