Helion

Rozwijaj i sprzedawaj aplikacje dla Windows 8!

JavaScript
Aplikacje dla **Windows 8**

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

telerik

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.

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.

Spis treści

11

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 8jest 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 przenosisz) 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 kontrolek sprawiających, że aplikacja będzie przyjazna w dotyku, a w razie potrzeby umożliwiała interakcją 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 toczonych 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 obsługi[1].

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 platformy[2]. Na szczęście, sama platforma oferuje wielką pomoc.

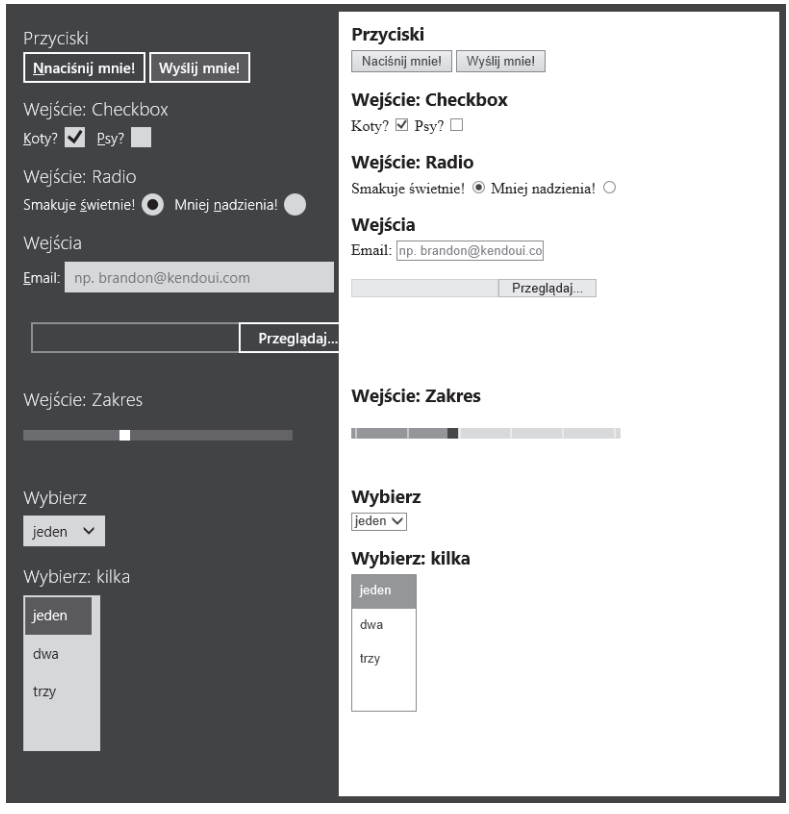
Przyjazne w dotyku kontrolki HTML

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 kontrolek 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 celem[3]. 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.):

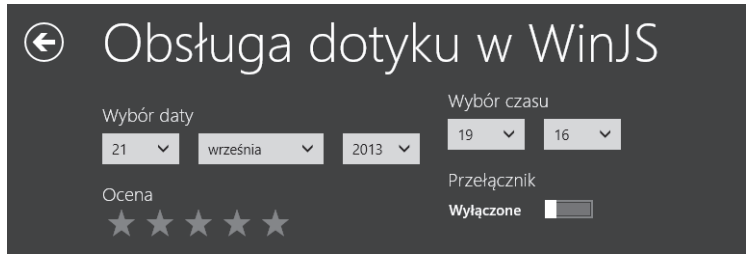
```
// ui-dark.cs
input[type=checkbox] {
  width: 21px;
  height: 21px;
  margin-right: 5px;
}
```

[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, odczują, jak platforma, a więc i Twoja aplikacja, są zaprojektowane z myślą o ich dłoniach. Możesz korzystać z przyjaznych w dotyku kontrolek za darmo, dołączając do swojej aplikacji jeden z domyślnych szablonów stylów (`ui-dark` lub `ui-light`). Zachęcam do używania ich i pozostania przy wielkościach kontrolek określonych przez platformę, chyba że masz naprawdę ważny powód, aby je zmienić[4].

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 kontrolek; 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.:

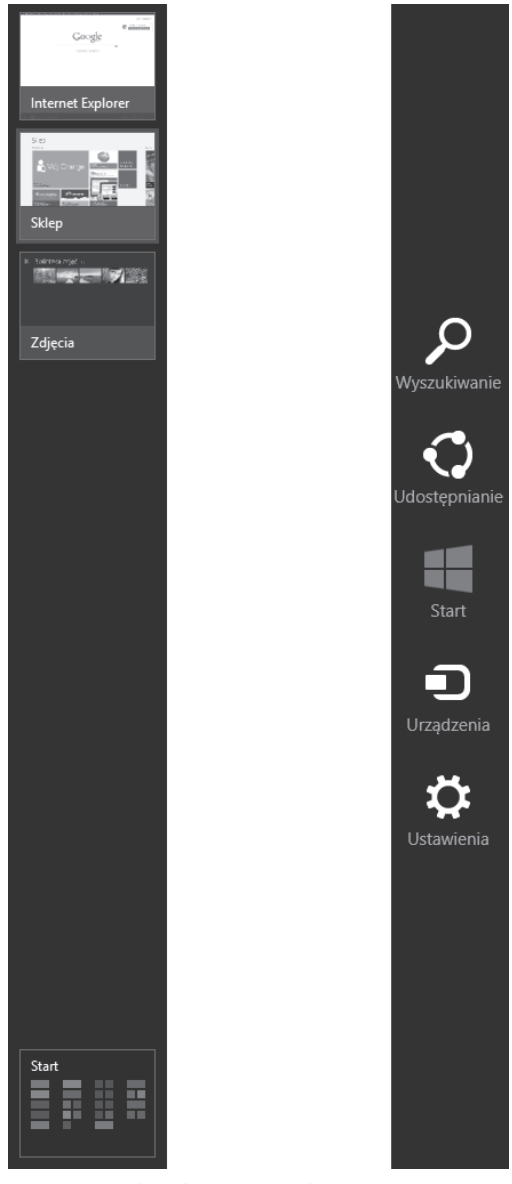
```
// ui-dark.css
.win-rating .win-star {
  -ms-flex: 1 1 auto;
  height:28px;
  width: 28px;
  padding: 0 6px;
  font-family: "Segoe UI Symbol";
  font-size: 28px;
  ...
}
```

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 kontrolek, 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://tinysells.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 dużo[5]. 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 brzegi[6]. 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 kontrolek 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.

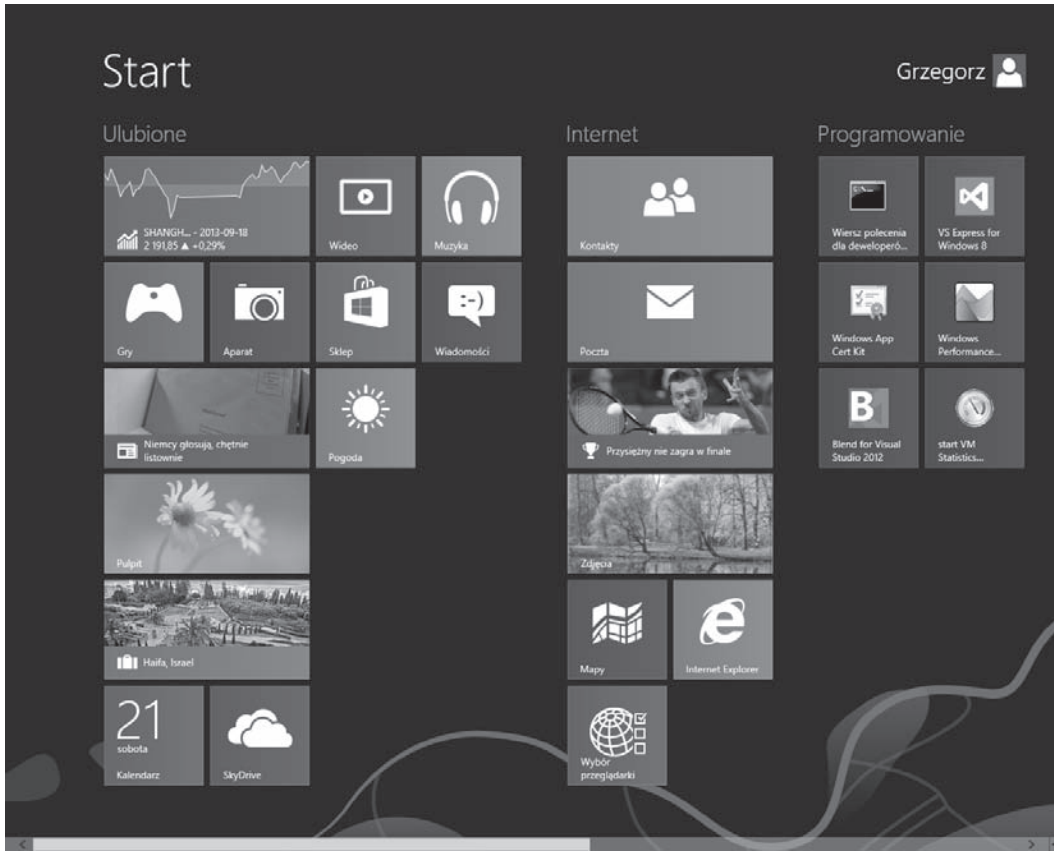
Podobnie jak wszystkich innych elementów opisanych wcześniej, również kontrolki `ListView` możesz używać za darmo w swoich aplikacjach w stylu Windows 8.

Rysunek 11.6. Wbudowana obsługa dotyku w kontrolce ListView

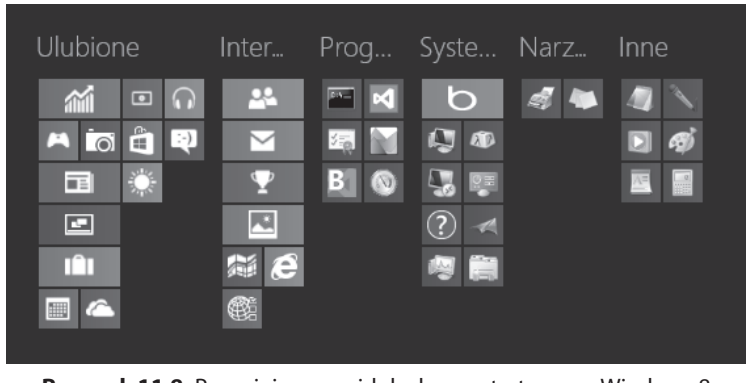
Inną kontrolką, znakomitą dzięki swojej wbudowanej obsłudze dotyku, jest `SemanticZoom`. Jest zbudowana na bazie kontrolki `ListView` i oferuje użytkownikowi, oprócz widoku pełnej listy, również możliwość przełączania pomiędzy wizualizacjami danych, np. według kategorii lub daty. Przypomina to filtrowanie, 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 szczypnięcie, *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 „krainę" 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; spójrzmy na przykład. Załóżmy, że dodaliśmy nowe pole z kategorią w galerii zdjęć z rozdziału 3., „Układ":



Rysunek 11.7. Ekran startowy Windows 8

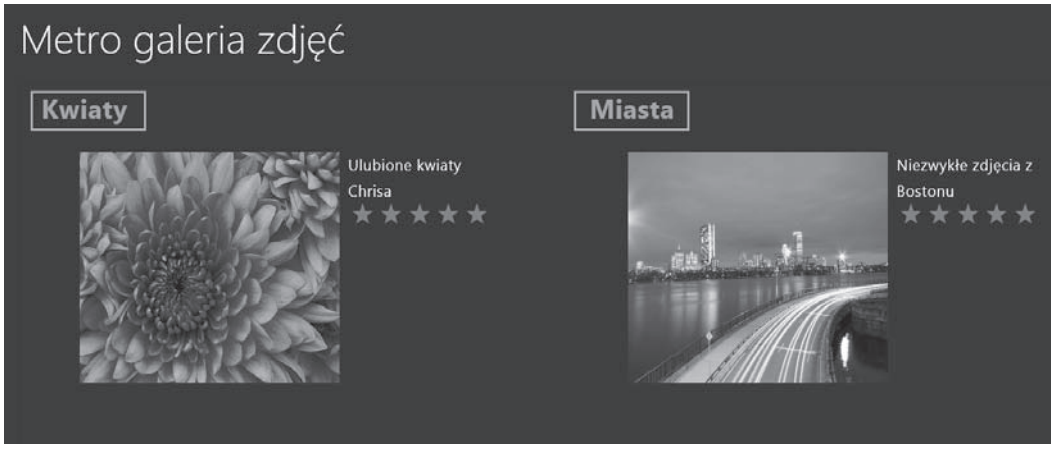


Rysunek 11.8. Pomniejszony widok ekranu startowego Windows 8

```
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 kontrolek `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:

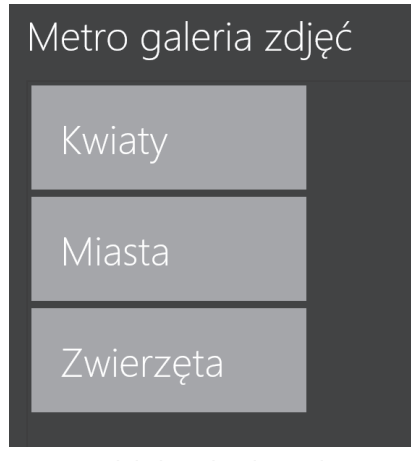
```
<div id="zoomTemplate" data-win-control="WinJS.Binding.Template">
  <div class="zoomItem">
    <h1 class="zoomItem-Text" data-win-bind="innerText: title"></h1>
  </div>
</div>
```

Pomniejszony widok składa się po prostu z tytułów kategorii, z kilkoma stylami CSS powiększającymi czcionkę 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 we 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 kontrolek `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 8[7].

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

Przede 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, począwszy 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.Enumeration.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 `MediaCaptureInitialization` ↪`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:

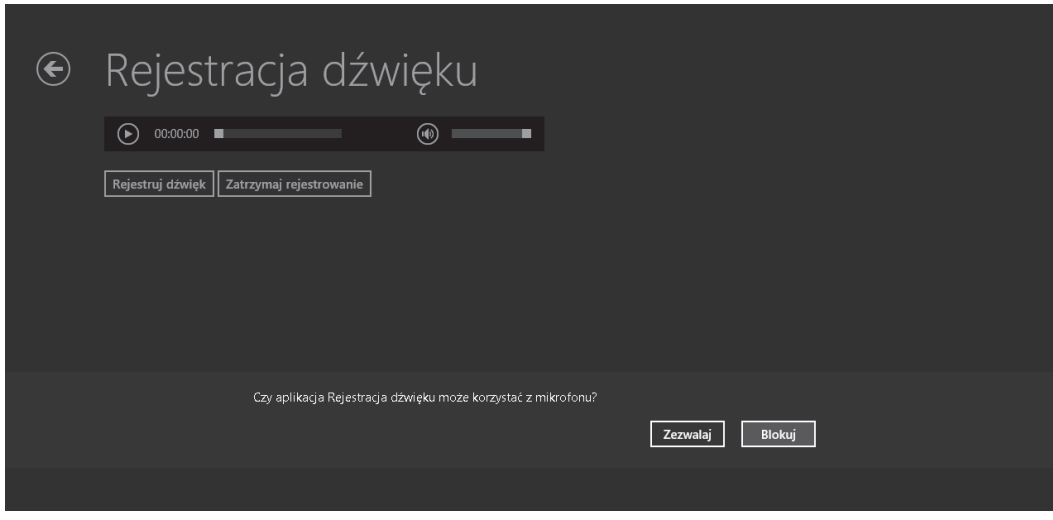
```
var profile;
var mediaProperties = Windows.Media.MediaProperties;
mediaCapture = new Windows.Media.Capture.MediaCapture();
mediaCapture.initializeAsync(captureSettings).done(function (result) {
  profile = mediaProperties.MediaEncodingProfile.createMp3(
    mediaProperties.AudioEncodingQuality.high);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

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 upewnisz 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 `'printaskrequested'`, 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]

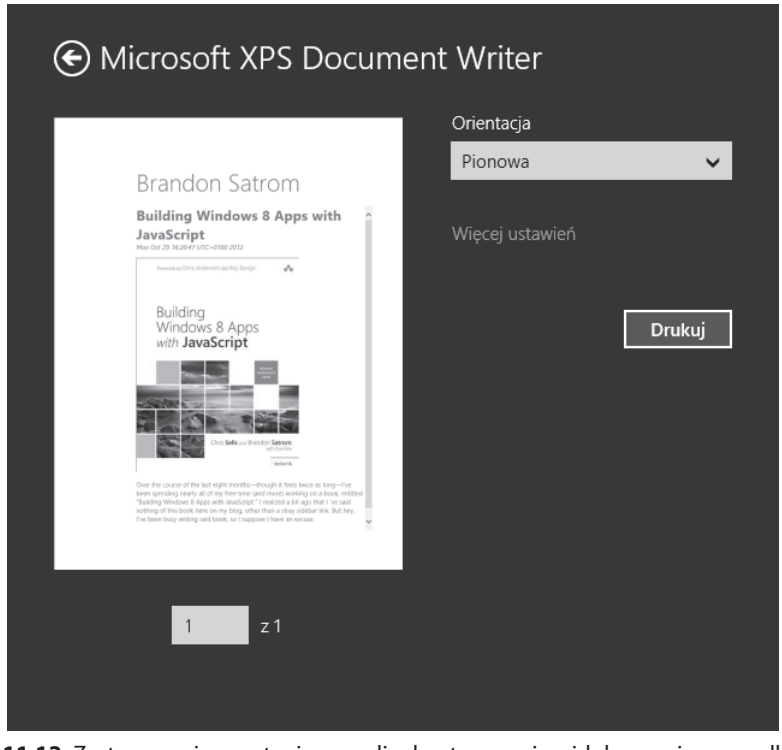
[8] Więcej informacji na temat tworzenia stylów CSS do drukowania znajduje się pod adresem *http://benfrain.com/create-print-styles-using-css3-media-queries/*.

```
@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 naprawę 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.

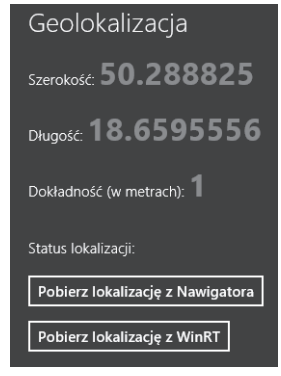
Użycie obiektu Geolocator

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 JavaScripcie

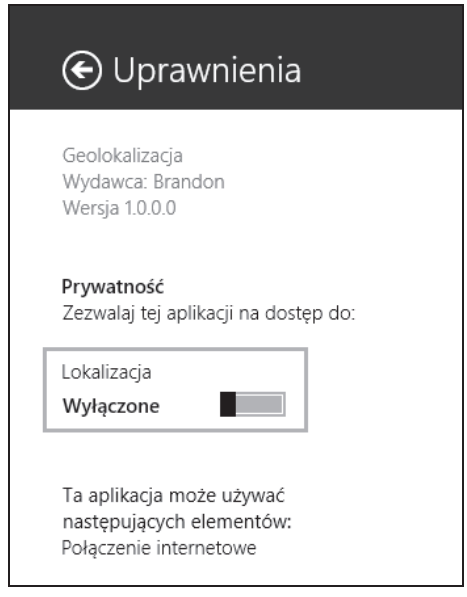
Obiekt `location` zwrócony przez konstruktor `Windows.Devices Geolocator` zawiera te same dane dotyczące długości, szerokości geograficznej i dokładności, jak obiekt nawigacji w przeglądarce, ale zawiera również właściwość `locationStatus`, której tam nie ma. Wartość `locationStatus` dostarcza nieco 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 „niedostępności 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 `Geolocator` środowiska WinRT oferują dostęp do regularnych aktualizacji. Do obiektu `Geolocator` musisz jedynie dodać nasłuch zdarzenia `'positionchanged'`:

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

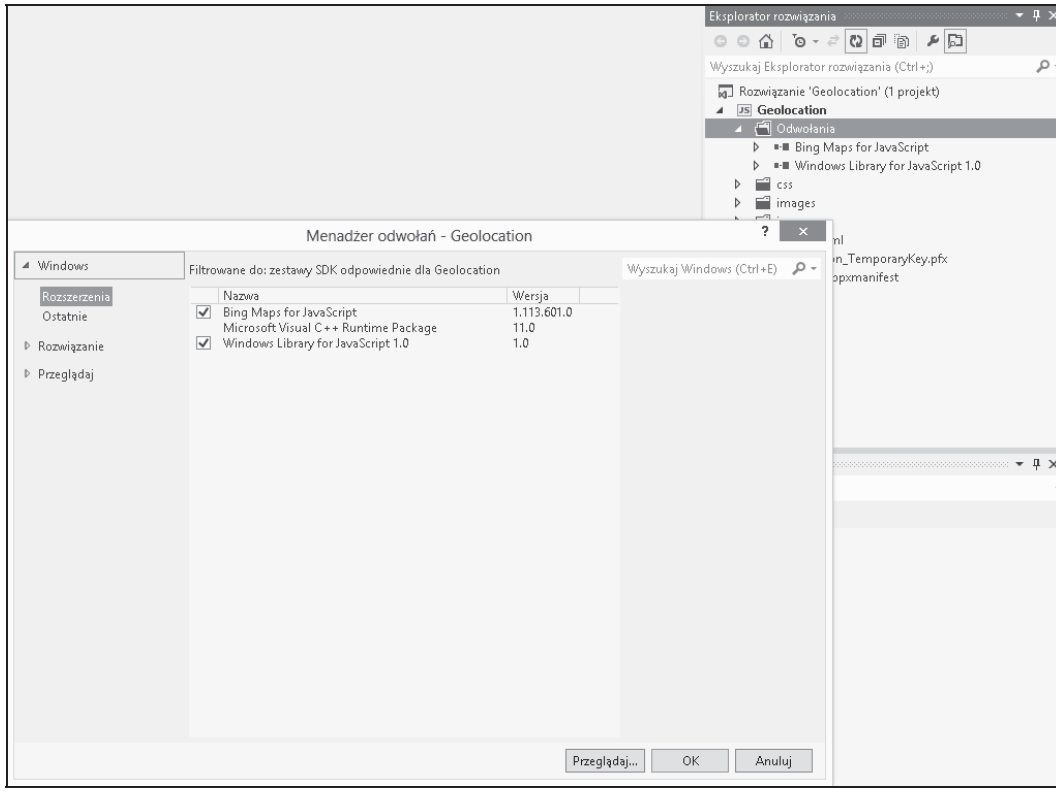
Tak jak we wszystkich procedurach obsługi zdarzeń, jeżeli nie potrzebujesz danych lokalizacyjnych, możesz usunąć nasłuch:

```
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 Studio[9]. 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

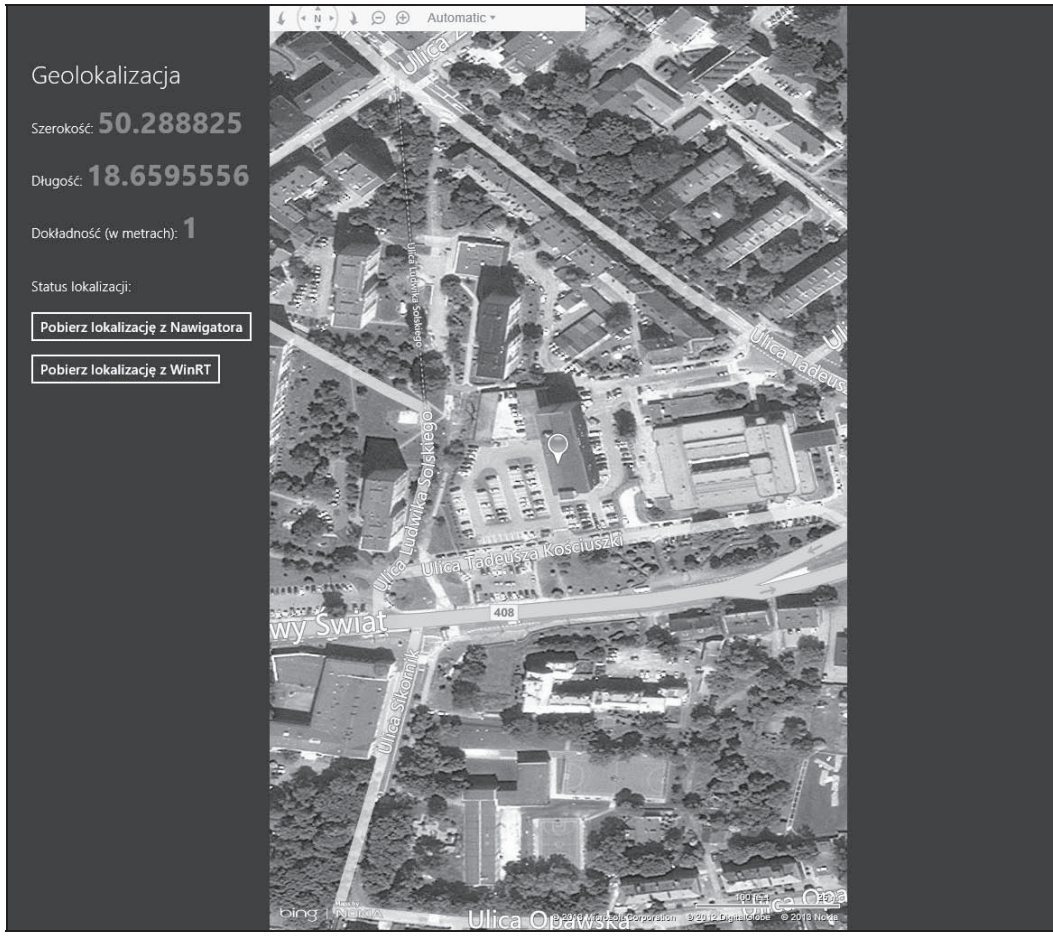
Podczas pracy z usługami lokalizacyjnymi często musimy przetestować większą liczbę lokalizacji, oprócz położenia domu lub miejsca pracy. Na szczęście, system Windows 8 oferuje funkcjonalność symulacji lokalizacji, dostępną za pomocą opcji *Ustaw lokalizację* w menu symulatora, przedstawionego na rysunku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

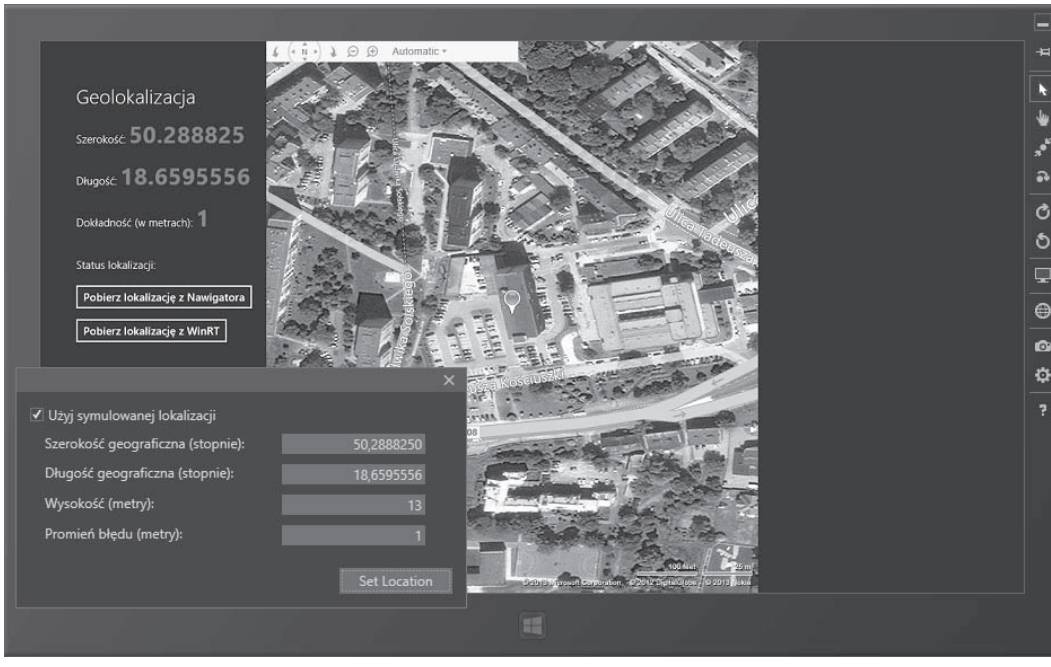


Rysunek 11.17. Zastosowanie usług lokalizacyjnych z pakietem Bing Maps SDK

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 (*pitch*, *roll*, *yaw*).
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 czujnika[10]. 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 drugiej[11]. Zacznijmy 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?

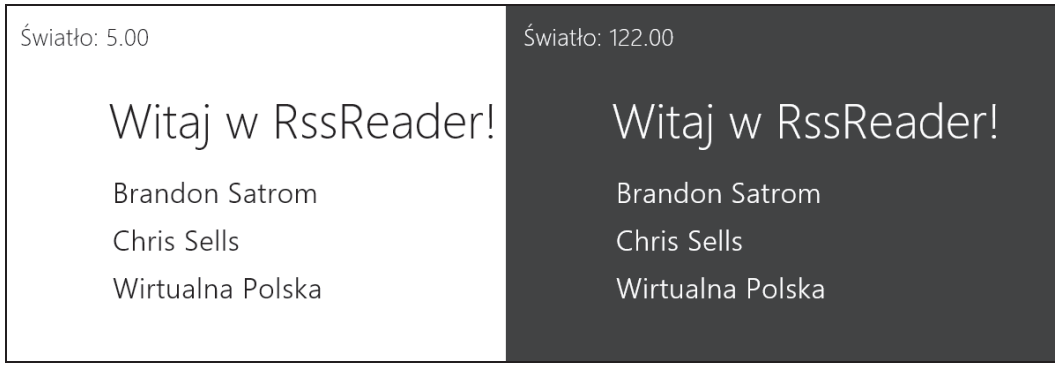
[12] Luks jest standardową jednostką oświetlenia. Służy do wyrażenia intensywności światła postrzeganej przez ludzkie oko. Jeżeli ten opis Ci nie wystarcza, możesz zajrzeć do Wikipedii pod adres *http://pl.wikipedia.org/wiki/Luks_(fotometria)* (*http://tinysells.com/215*).

```
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 przyspieszeniomierzem

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.innerText = "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 potrząsanie 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").innerText = "POTRZĄŚNIĘCIE! Odświeżanie...";
    syn.retrieveFeedAsync(url)
      .then(processPosts, downloadError)
      .done(function () {
        element.querySelector(".pagetitle").innerText = options.feed.title;
  });
});

function processPosts(request) { … }
```

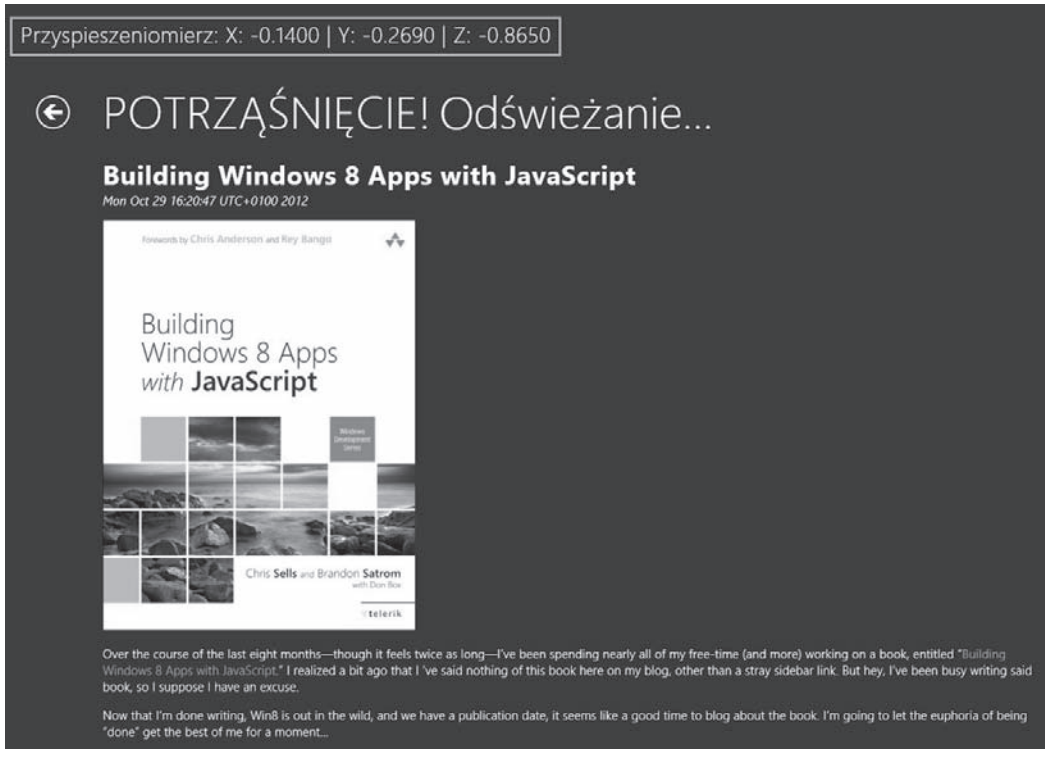
Kiedy pojawi się zdarzenie `shake`, zmienimy na chwilę tytuł strony, aby poinformować użytkownika, że trwa odświeżanie, po czym wywołamy klienta syndykacji w celu pobrania zestawu aktualnych wpisów z kanału. Rezultat porządnego potrząśnięcia urządzeniem przedstawiam na rysunku 11.20.

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 rzeczywistą[13]. Aby zobaczyć kompas w akcji, zbudujmy... 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 przyspieszeniomierza

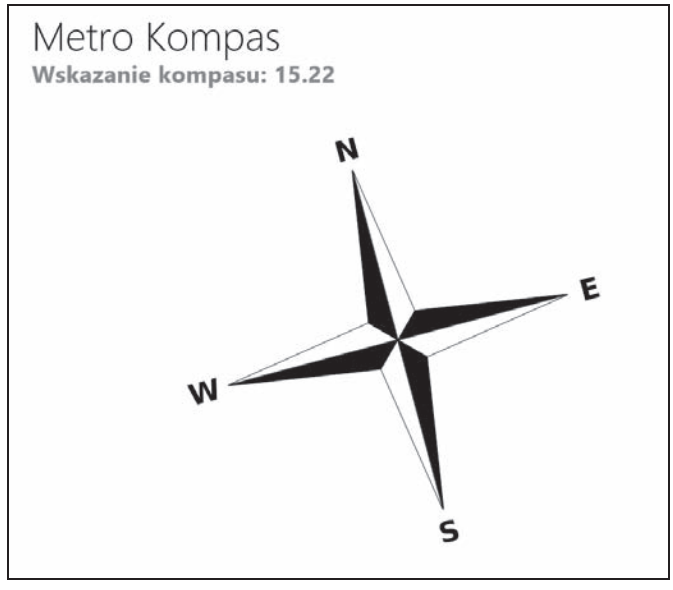
Tak jak w innych czujnikach, pobierzemy instancję obiektu `Compass` za pomocą metody `getCurrent`, a następnie skonfigurujemy nasłuch odpowiadający na zmianę kierunku:

```
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, począwszy 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

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

Ś

T

U

V

W

Z

Ż

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/nowosci

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

Informatyka w najlepszym wydaniu

cena 77,00 zł

ISBN 978-83-246-7564-7

9 788324 675647

PEARSON