Contents at a Glance

Introduction .............................................................................................................. 1
1  Building Windows Store Apps ........................................................................ 5
2  WinJS Fundamentals ....................................................................................... 45
3  Observables, Bindings, and Templates .......................................................... 81
4  Using WinJS Controls .................................................................................... 113
5  Creating Forms ................................................................................................ 149
6  Menus and Flyouts .......................................................................................... 169
7  Using the ItemContainer, Repeater, and FlipView Controls ......................... 197
8  Using the ListView Control ............................................................................ 223
9  Creating Data Sources ..................................................................................... 263
10 Storing Data with Windows Azure ................................................................. 295
11 App Events and States .................................................................................... 311
12 Page Fragments and Navigation ..................................................................... 333
13 Creating Share and Search Contracts ............................................................. 353
14 Using the Live Connect API ........................................................................... 387
15 Graphics and Games ....................................................................................... 417
16 Creating a Task List App ................................................................................ 431
Index ..................................................................................................................... 449
# Table of Contents

**Introduction**
- Updated for Windows 8.1 .......................................................... 2  
- Prerequisites for This Book ...................................................... 3  
- Source Code ............................................................................. 3

1 **Building Windows Store Apps** .............................................. 5  
   - What Is a Windows Store App? .................................................. 5  
     - Microsoft Design Style Principles ....................................... 6  
     - Common Features of Windows Store Apps .......................... 7  
   - Creating Your First Windows Store App ................................. 13  
     - Creating the Visual Studio Project ..................................... 14  
     - Declaring App Capabilities ................................................. 15  
     - Creating the HTML Page .................................................... 17  
     - Creating the Style Sheet ..................................................... 18  
     - Creating the JavaScript File ................................................. 18  
     - Running the App ................................................................ 21  
   - Elements of a Windows Store App ........................................... 21  
     - JavaScript ........................................................................ 21  
     - HTML5 ............................................................................ 22  
     - Cascading Style Sheets 3 ..................................................... 22  
     - Windows Runtime ............................................................... 23  
     - Windows Library for JavaScript ......................................... 23  
     - What About jQuery? .......................................................... 24  
   - Building Windows Store Apps with Visual Studio .................. 26  
     - Windows Store App Project Templates ................................. 27  
     - Running a Windows Store App ............................................. 31  
   - Debugging a Windows Store App ............................................. 33  
     - Using the Visual Studio JavaScript Console Window .......... 33  
     - Setting Breakpoints ............................................................ 34  
     - Using the DOM Explorer ..................................................... 35  
   - Publishing to the Windows Store ............................................ 36  
     - Register as a Windows Developer ....................................... 36  
     - Submitting Your App .......................................................... 37  
     - Passing App Certification .................................................... 39  
   - Migrating from Windows 8 to Windows 8.1 ............................ 41  
   - Summary ............................................................................ 42
Contents

Using HTML5 Input Elements ............................................................ 155
Labeling Form Fields ......................................................................... 157
Entering a Number .............................................................................. 158
Entering a Value from a Range of Values ......................................... 159
Entering Email Addresses, URLs, Telephone Numbers, and Search Terms .......................................................... 160
Entering a Value from a List of Values .............................................. 162
Selecting Files .................................................................................. 162
Creating a Rich Text Editor ................................................................ 164
Displaying Progress .......................................................................... 165
Summary .......................................................................................... 167

6 Menus and Flyouts ...................................................................... 169
Using the Flyout Control ..................................................................... 169
Using the Menu Control ..................................................................... 172
Using the AppBar Control ................................................................. 176
Creating a Simple App Bar .................................................................. 176
Using App Bar Commands ..................................................................... 178
Showing Contextual Commands .......................................................... 181
Using the NavBar Control ................................................................. 184
Creating a Simple Nav Bar ................................................................. 184
Configuring App Settings .................................................................... 186
Creating About Page Settings ............................................................. 187
Creating Personal Settings ................................................................. 189
Displaying Windows Dialogs ............................................................... 192
Summary .......................................................................................... 195

7 Using the ItemContainer, Repeater, and FlipView Controls .......... 197
Using the ItemContainer Control ........................................................ 197
Styling an ItemContainer ..................................................................... 198
Interacting with an ItemContainer ....................................................... 200
Selecting an ItemContainer ................................................................. 202
Creating Drag-and-Drop Items ............................................................ 204
Using the Repeater Control ................................................................. 208
Using an External Template ............................................................... 210
Using a Nested Template ..................................................................... 211
Using the Repeater with the ItemContainer ......................................... 214
Using the FlipView Control ................................................................. 215
Displaying Page Numbers ................................................................. 219
Creating Custom FlipView Buttons ...................................................... 221
Summary .......................................................................................... 222
8 Using the ListView Control

Introduction to the ListView Control........................................................................224
Using Different ListView Layouts...........................................................................228
  Using Grid Layout...................................................................................................229
  Using List Layout...................................................................................................231
  Using Cell Spanning Layout...................................................................................231
Invoking Items in a ListView Control.........................................................................236
Selecting Items in a ListView Control.........................................................................238
Sorting Items in a ListView Control...........................................................................241
Filtering Items in a ListView Control..........................................................................242
Grouping Items in a ListView Control.........................................................................245
Switching Views with Semantic Zoom.........................................................................248
Switching a ListView Template Dynamically..............................................................253
Using Drag and Drop..................................................................................................256
  Reordering Items in a ListView...............................................................................256
  Dragging Items from ListViews.................................................................................258
Summary....................................................................................................................262

9 Creating Data Sources

Creating Custom Data Sources....................................................................................263
  Creating the Data Source Class..............................................................................264
  Creating a Data Adapter..........................................................................................264
  Implementing the getCount() Method......................................................................265
  Implementing the itemsFromIndex() Method..........................................................265
  Implementing the insertAtEnd() Method...................................................................267
  Implementing the remove() Method.........................................................................267
  Implementing the change() Method.........................................................................268
  Handling Errors........................................................................................................268
  Implementing the setNotificationHandler() Method..............................................269
Creating a File Data Source........................................................................................270
  Using the File Data Source......................................................................................271
Creating a Web Service Data Source...........................................................................276
  Creating the Data Source........................................................................................276
  Creating the Web Service........................................................................................278
  Using the Web Service Data Source.........................................................................280
Creating an IndexedDB Data Source............................................................................281
  Overview of IndexedDB..........................................................................................282
  Using the IndexedDB Data Source..........................................................................286
Summary....................................................................................................................293
## Contents

### 10 Using Windows Azure Mobile Services

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating a Mobile Service</td>
<td>295</td>
</tr>
<tr>
<td>Creating a Database Table</td>
<td>297</td>
</tr>
<tr>
<td>Installing the Mobile Services for WinJS Library</td>
<td>298</td>
</tr>
<tr>
<td>Performing Inserts, Updates, and Deletes</td>
<td>299</td>
</tr>
<tr>
<td>Connecting to the Remote Database Table</td>
<td>299</td>
</tr>
<tr>
<td>Inserting Database Data</td>
<td>299</td>
</tr>
<tr>
<td>Updading Database Data</td>
<td>300</td>
</tr>
<tr>
<td>Deleting Database Data</td>
<td>301</td>
</tr>
<tr>
<td>Performing Database Queries</td>
<td>301</td>
</tr>
<tr>
<td>Looking Up a Single Database Record</td>
<td>301</td>
</tr>
<tr>
<td>Retrieving a Set of Database Records</td>
<td>302</td>
</tr>
<tr>
<td>Performing Validation</td>
<td>304</td>
</tr>
<tr>
<td>Performing Custom Actions</td>
<td>306</td>
</tr>
<tr>
<td>Debugging Script Errors</td>
<td>308</td>
</tr>
<tr>
<td>Summary</td>
<td>309</td>
</tr>
</tbody>
</table>

### 11 App Events and States

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>App Events</td>
<td>311</td>
</tr>
<tr>
<td>Handling the Activated Event</td>
<td>311</td>
</tr>
<tr>
<td>Handling the Error Event</td>
<td>312</td>
</tr>
<tr>
<td>Deferring Events with Promises</td>
<td>313</td>
</tr>
<tr>
<td>Creating Custom Events</td>
<td>314</td>
</tr>
<tr>
<td>Suspending, Terminating, and Resuming an App</td>
<td>315</td>
</tr>
<tr>
<td>Detecting When an App Is Suspended and Terminated</td>
<td>316</td>
</tr>
<tr>
<td>Detecting the Previous Execution State</td>
<td>316</td>
</tr>
<tr>
<td>Testing Application State with Visual Studio</td>
<td>317</td>
</tr>
<tr>
<td>Storing State with Session State</td>
<td>318</td>
</tr>
<tr>
<td>Designing for Different Window Sizes</td>
<td>320</td>
</tr>
<tr>
<td>Setting the Minimum App Width</td>
<td>320</td>
</tr>
<tr>
<td>Using CSS Media Queries</td>
<td>321</td>
</tr>
<tr>
<td>Using the window resize Event</td>
<td>324</td>
</tr>
<tr>
<td>Scaling Content to Fit Different Screen Resolutions</td>
<td>326</td>
</tr>
<tr>
<td>Defining a Viewport</td>
<td>326</td>
</tr>
<tr>
<td>Using the ViewBox Control</td>
<td>329</td>
</tr>
<tr>
<td>Summary</td>
<td>332</td>
</tr>
</tbody>
</table>

### 12 Page Fragments and Navigation

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using the HtmlControl Control</td>
<td>333</td>
</tr>
<tr>
<td>Creating a Page Control</td>
<td>336</td>
</tr>
</tbody>
</table>
Creating Multi-Page Apps .......................................................... 340
Creating a Navigation App ....................................................... 340
Understanding the Navigation App default.html Page ............ 341
Adding New Page Controls to a Navigation App ....................... 343
Navigating to Another Page ..................................................... 345
Understanding the Navigation API .......................................... 346
Understanding the PageControlNavigator Control ................. 347
Understanding Navigation State ............................................. 347
Summary .................................................................................. 351

13 Creating Share and Search Contracts 353
Supporting Sharing ................................................................. 354
Creating a Share Source .......................................................... 356
Creating a Share Target ........................................................... 360
Using the Search Charm .......................................................... 368
Declaring Your App as a Search Provider ................................. 369
Providing Search Suggestions ................................................... 370
Handling Search Activation ..................................................... 372
Adding a Search Results Page .................................................. 373
Using the SearchBox Control ................................................... 376
Adding the SearchBox Control to a Page ................................. 377
Providing Search Suggestions ................................................... 378
Displaying Search Results ....................................................... 379
Using the Windows Content Indexer ........................................ 380
Understanding the Windows Content Indexer API .................... 381
Creating an Indexer Helper ...................................................... 381
Using the Indexer Helper ........................................................ 382
Summary .................................................................................. 385

14 Using the Live Connect API 387
Installing the Live SDK ............................................................ 388
Adding a Reference to the Live SDK ........................................ 388
Registering Your App ............................................................. 389
Initializing the Live Connect SDK ............................................ 391
Specifying Different Scopes .................................................... 391
Authenticating a User ............................................................. 394
Logging a User into Live Connect ............................................. 394
Creating Account Settings ....................................................... 396
Authentication and Windows Azure Mobile Services ............... 401
Configuring Your Mobile Service ............................................. 401
Setting Permissions for Your Mobile Service ......................... 402
Updating the Mobile Server Scripts ........................................ 402
Logging Into Azure Mobile Services ...................................... 404
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieving Basic User Information</td>
<td>406</td>
</tr>
<tr>
<td>Uploading and Downloading Files from SkyDrive</td>
<td>408</td>
</tr>
<tr>
<td>Listing SkyDrive Folders and Files</td>
<td>409</td>
</tr>
<tr>
<td>Downloading Files from SkyDrive</td>
<td>411</td>
</tr>
<tr>
<td>Uploading Files to SkyDrive</td>
<td>413</td>
</tr>
<tr>
<td>Summary</td>
<td>415</td>
</tr>
</tbody>
</table>

### 15 Graphics and Games

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of the Game</td>
<td>418</td>
</tr>
<tr>
<td>Creating the Game Tiles</td>
<td>419</td>
</tr>
<tr>
<td>Playing the Game Sounds</td>
<td>420</td>
</tr>
<tr>
<td>Creating the Game Canvas</td>
<td>421</td>
</tr>
<tr>
<td>Capturing User Interaction</td>
<td>424</td>
</tr>
<tr>
<td>Creating the Update Loop</td>
<td>425</td>
</tr>
<tr>
<td>Creating the Render Loop</td>
<td>427</td>
</tr>
<tr>
<td>Summary</td>
<td>429</td>
</tr>
</tbody>
</table>

### 16 Creating a Task List App

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of the App</td>
<td>432</td>
</tr>
<tr>
<td>Setting Up the App</td>
<td>433</td>
</tr>
<tr>
<td>Connecting to External Services</td>
<td>435</td>
</tr>
<tr>
<td>Optimistic Inserts, Updates, and Deletes</td>
<td>437</td>
</tr>
<tr>
<td>Adapting to Screen Changes</td>
<td>440</td>
</tr>
<tr>
<td>Creating a Custom Control</td>
<td>444</td>
</tr>
<tr>
<td>Using Text to Speech</td>
<td>446</td>
</tr>
<tr>
<td>Summary</td>
<td>448</td>
</tr>
</tbody>
</table>

| Index                                                                  | 449  |
About the Author

Formerly a Senior Program Manager at Microsoft, Stephen Walther now runs his own consulting and training company www.SuperexpertTraining.com. He flies to companies and provides hands-on training on building Windows Store apps.

Stephen was completing his Ph.D. at MIT and teaching classes on metaphysics at MIT and Harvard when he abruptly realized that there is no money in metaphysics. He dropped out to help found two successful Internet startups. He created the Collegescape website, a website used by more than 200 colleges, including Stanford, Harvard, and MIT, for online college applications (sold to ETS). He also was a founder of CityAuction, which was one of the first and largest auction websites (sold to CitySearch).
Dedication

This book is dedicated to Jon Robert Walther, who is a Jedi ninja.

Acknowledgments

Yikes, it takes too much work to write a technical book—don't ever do it! I would like to blame my editor Neil Rowe for talking me into writing another book. I also want to blame my wife Ruth Walther for failing to talk me out of it. Finally, I want to blame my technical editors Jeff Burtoft and James Boddie for doing such a careful job of coming up with ways to improve the book and forcing me to spend even more time working on the book.
We Want to Hear from You!

As the reader of this book, you are our most important critic and commentator. We value your opinion and want to know what we’re doing right, what we could do better, what areas you’d like to see us publish in, and any other words of wisdom you’re willing to pass our way.

You can email or write me directly to let me know what you did or didn’t like about this book—as well as what we can do to make our books stronger.

Please note that I cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail I receive, I might not be able to reply to every message.

When you write, please be sure to include this book’s title and author as well as your name and phone or email address. I will carefully review your comments and share them with the author and editors who worked on the book.

Email:  feedback@samspublishing.com

Mail:  Neil Rowe
       Executive Editor
       Sams Publishing
       800 East 96th Street
       Indianapolis, IN  46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.
Introduction

If you want to build a software application and reach the largest possible market of customers and make the most money then it makes sense for you to build a Windows 8.1 app.

Microsoft Windows is the most popular operating system in the world. Windows accounts for more than 90 percent of the operating system market. More than 100 million licenses for Windows 8 were sold in its first six months of release. The size of the Windows market dwarfs the size of every other marketplace for software applications (including the iPhone and Android markets).

I want to own a toilet made of solid gold, Nathan Myhrvold’s jet, and a Tesla Roadster (orange). These are modest goals, and I know that many of you reading this book share the same goals. The most likely way for you or me (hopefully me) to reach these goals is to build Windows 8.1 apps.

When you build a Windows 8.1 app, you can sell your app right within Windows 8.1 itself. Windows 8.1 includes the Windows Store (shown in Figure I.1) where you can list your app for anywhere between free and $999.99. You can sell a variety of different types of apps including productivity apps (think task lists and time trackers) and games (think Angry Birds and Cut the Rope).

FIGURE I.1 You can sell your app in the Windows Store.
This book is all about building Windows apps that you can sell in the Windows Store. In particular, you learn how to build Windows apps using JavaScript and HTML5.

Why JavaScript and HTML5? You can build Windows apps using other technologies such as C# and XAML or C++, but this book focuses exclusively on building Windows apps with JavaScript and HTML5.

The advantage of building Windows apps with JavaScript and HTML5 is that you can leverage your existing skills building websites to build Windows applications. If you are already comfortable programming with JavaScript, HTML, and Cascading Style Sheets then you should find it easy to build Windows apps.

This book covers everything you need to know to build Windows apps. You learn how to use the Windows Library for JavaScript (WinJS) to create JavaScript applications. In particular, you learn how to use WinJS controls such as the Rating, Menu, Repeater, and ListView controls.

You also learn how to work with the Windows Runtime. By taking advantage of the Windows Runtime, you can access Windows 8.1 functionality to do things that you could not normally do in a pure web app, such as capture video and sound and convert text to speech.

By the end of this book, you will understand how to create Windows apps, such as game apps and productivity apps. In Chapter 15, “Graphics and Games,” you learn how to create a simple arcade game—the Brain Eaters game. And, in Chapter 16, “Creating a Task List App,” you learn how to build a productivity app—the MyTasks app.

Read this book, build a Windows app, sell lots of copies, and buy a jet.

**Updated for Windows 8.1**

This book has been extensively updated for Windows 8.1. Changes have been made to every chapter. All of the code associated with this book has been reviewed and updated to be compatible with Windows 8.1.

Windows 8.1 includes several important new controls, including the Hub, Repeater, ItemContainer, SearchBox, WebView, and NavBar controls. This book covers all of these new controls in depth.

Windows 8.1 ships with a new version of the Windows Library for JavaScript (WinJS 2.0). This new version has significant new features such as the WinJS Scheduler. I discuss the new WinJS Scheduler in Chapter 2, “WinJS Fundamentals.”

Windows 8.1 includes important backwards breaking changes. Unlike Windows 8, Windows 8.1 no longer supports discrete view states such as a snapped or filled state. I discuss these changes in Chapter 11, “App Events and States.”

Finally, I added four new chapters to this book. I added a chapter that covers the new ItemContainer and Repeater controls (Chapter 7, “Using the ItemContainer, Repeater, and FlipView Controls”), a chapter devoted to using Windows Azure Mobile Services (Chapter 10, “Storing Data with Windows Azure”), a chapter on implementing share and
search (Chapter 13, “Creating Share and Search Contracts”), and a chapter on building a productivity app (Chapter 16).

Prerequisites for This Book

If you can build a website using JavaScript, HTML, and Cascading Style Sheets then you have the skills that you need to read and understand this book.

There are two software requirements for building Windows apps and using the code from this book.

First, you must build a Windows 8.1 app on the Windows 8.1 operating system. Let me repeat this: You must have Windows 8.1 installed on your computer to use the code from this book.

Second, in order to use the code from this book, you need Microsoft Visual Studio 2013. There is a free version of Visual Studio 2013—Microsoft Visual Studio Express 2013 for Windows—which you can download from the Microsoft.com website.

Source Code

You can download all of the source code associated with this book from GitHub: https://github.com/StephenWalther/Windows8.1AppsUnleashed

Click the Downloads link to download the latest version of the code in a zip file.
In this chapter, I introduce you to the basics of building Windows Store apps. I start off by explaining how a Windows Store app differs from a traditional Windows desktop application. You learn what makes a Windows Store app a Windows Store app.

Feeling fearless and bold, and hoping that you too feel fearless and bold, I next guide you through building your first Windows store app. You learn how to take advantage of the features of Microsoft Visual Studio 2013 to build, run, and debug a Windows Store app.

Next, we dive into a discussion of the fundamental elements of a Windows Store app. You learn how a Windows Store app is forged out of HTML5, JavaScript, the Windows Library for JavaScript, and the Windows Runtime.

Finally, we get to the money part. I explain how you can publish your Windows Store app to the Windows Store and start collecting those dollars.

What Is a Windows Store App?

I can still remember the first time that I used an iPhone. When you scroll the screen on an iPhone, the screen actually bounces! And when you add an email to the trash, the email gets sucked into the trashcan! It’s as if there is a little universe inside an iPhone and it follows our physical laws.

For some reason—that I have not explored and that I do not completely understand—this illusion that there is a second universe inside my iPhone makes me happy. It makes interacting with an iPhone fun.
Now we come to Windows. Except for the dancing card thing in Windows Solitaire, I can’t think of anything in Windows that has ever created this same sense of fun. I can’t remember the last time that Windows made me laugh or brought me joy.

With Windows Store apps, Microsoft has finally acknowledged that user experience matters—in a big way. The heart of Windows Store apps is a set of user experience principles named the Microsoft design style principles. By embracing the Microsoft design style principles, you can create Windows Store apps that seem more alive and that are a pleasure to use.

**Microsoft Design Style Principles**

The Microsoft design style principles is a set of user experience design principles developed by Microsoft in the context of building the Windows Phone, Xbox Live, and the (now defunct) Zune. You also can see the Microsoft design principles applied to Microsoft websites such as Microsoft SkyDrive and the Windows Azure Portal. Get ready. Here they are:

1. **Show pride in craftsmanship**
   - Devote time and energy to small things that are seen often by many.
   - Engineer the experience to be complete and polished at every stage.

2. **Do more with less**
   - Solve for distractions, not discoverability. Let people be immersed in what they love and they will explore the rest.
   - Create a clean and purposeful experience by leaving only the most relevant elements on screen so people can be immersed in the content.

3. **Be fast and fluid**
   - Let people interact directly with content, and respond to actions quickly with matching energy.
   - Bring life to the experience, create a sense of continuity and tell a story through meaningful use of motion.

4. **Be authentically digital**
   - Take full advantage of the digital medium. Remove physical boundaries to create experiences that are more efficient and effortless than reality.
   - Embrace the fact that we are pixels on a screen. Design with bold, vibrant and crisp colors and images that go beyond the limits of real-world material.
What Is a Windows Store App?

Common Features of Windows Store Apps

Windows Store apps are applications that follow the Microsoft design style principles. Furthermore, Windows Store apps are designed to run on the Windows 8 or Windows RT operating system.

5. Win as one

- Leverage the ecosystem and work together with other apps, devices and the system to complete scenarios for people.
- Fit into the UI model to reduce redundancy. Take advantage of what people already know to provide a sense of familiarity, control, and confidence.

NOTE

The Microsoft design style principles were originally known as Metro design principles. This list of Microsoft design style principles was taken from http://msdn.microsoft.com/en-us/library/windows/apps/hh464920 and http://msdn.microsoft.com/en-us/library/windows/apps/hh465424.aspx.

When I first read these principles, my initial reaction was that they seemed overly abstract and squishy. Exactly the type of principles that would be created by beret-wearing user experience guys.

But then, when I saw how the principles were applied in practice—when building actual Windows Store apps—I started to develop a better appreciation for these principles.

Take the “Do more with less” design principle. One of the distinctive features of a Windows Store app is the lack of chrome. Ironically, a Windows Store app is a Windows app without the window. Windows Store apps are full-screen apps.

This lack of chrome makes it easier to concentrate on the content of the application. For example, Windows 8 includes two version of Internet Explorer: a desktop version and a full-throated Windows 8 version that follows the Microsoft design style principles.

I really prefer using the Windows 8 version of Internet Explorer over the desktop version. When using the Windows 8 version, all you see is the web page, which is the point of the application in the first place.

Or consider the “Be fast and fluid” principle. The reason that I like interacting with my iPhone so much is the illusion of motion, and this illusion is created by the judicious use of animations: On an iPhone, objects bounce and wobble.

When building a Windows Store app, you are encouraged to take advantage of animations. For example, if you use the standard ListView control—which we discuss in detail later in this book—then you get animations when you add or remove items. When you add an item to a ListView, it not only appears, it glides into place. When you remove an item, it doesn’t just disappear, items above and below it collapse into place.
All Windows Store apps have a common set of features. Let me explain these features by pointing them out in the context of the Bing News app that’s included with Windows 8.

**NOTE**
It is worth pointing out that the standard Windows 8 Bing News app discussed in this section was written using HTML5 and JavaScript (using the same techniques described in this book). In case you are curious, you can view the HTML and JavaScript source for the News app by opening the hidden folder where Windows apps are installed located at `Program Files\WindowsApps`.

**Support for Keyboard, Mouse, Touch, and Stylus**
One of the most distinctive characteristics of a Windows Store app is its oversized tiles and buttons and generous use of whitespace. All of this user interface (UI) roominess makes Windows Store apps friendly to fat fingers.

Windows Store apps are designed to work equally well when used on a touch-only tablet and when used on a desktop computer with a keyboard and mouse. Windows Store apps are designed to be gropeable.

The nice thing about how Windows 8 works is that you don’t need to put a lot of thought into supporting touch as a developer. As long as you stick with the standard WinJS controls, you get both keyboard and touch support for free.

**Using the App Bar and Nav Bar**
Figure 1.1 contains a screenshot of the Windows 8 Bing News app with the home page of Fox News open. Notice that the only thing that you see is the content of Fox News. No toolbars, no menus, no status bars.

![FIGURE 1.1 Windows 8 Bing News app](image)
In a Windows Store app, you hide all of your commands in the app bar. The app bar appears only when you swipe from the bottom or top of the screen or you right-click the screen.

The app bar for the Bing News app includes commands such as Pin to Start, Refresh, and Help. You can see the app bar at the bottom of Figure 1.2.

Notice in Figure 1.2 that there is another bar at the top of the screen. This bar is called the nav bar and you use it to navigate. In the case of the Bing News app, the nav bar enables you to navigate to different news sources such as the *Wall Street Journal*, Fox News, and the *New York Times*.

**Using Charms**

If you swipe from the right edge of the screen or mouse to either of the right corners or press the keyboard combination Win+C then the charms are revealed (see Figure 1.3).
Here’s a list of the standard charms:

▶ Search—Enables you to search content in the current app and other apps
▶ Share—Enables you to share content in the current app with other apps
▶ Start—Navigates you to the Start screen
▶ Devices—Enables you to connect to a device
▶ Settings—Enables you to configure both app settings and system settings

These charms provide you with standard locations to place common application functionality. For example, all Windows Store app settings should appear in the Settings charm (see Figure 1.4). This makes it much easier for users to find your settings.
What Is a Windows Store App?

When you are building a Windows Store app, you don’t build your own Settings menu. Instead, you extend the Settings charm with your custom app settings. I discuss the details of doing this in Chapter 6, “Menus and Flyouts.”

**Different App Sizes and Orientations**

Every Windows 8.1 app supports a minimum width of either 500 pixels or 320 pixels. For example, if a Windows 8.1 app has a minimum horizontal size of 500 pixels then the app can be resized to any size between 500 pixels and the maximum screen size of the device where the app is displayed.

If you are lucky enough to have a sufficiently large screen, then you can display multiple running apps side by side (up to four apps per monitor). For example, Figure 1.5 illustrates three Windows 8.1 apps running side by side (the Calendar, Maps, and News apps).

**WARNING**

You cannot display more than two 500 pixel apps on a 1,024 pixel by 768 pixel screen because that would violate the laws of mathematics.
A Windows Store app also must work when used with different device orientations. For example, when an app is viewed on a tablet computer, the user always has the option of rotating your app from a landscape to a portrait orientation.

When building Windows Store apps, you need to design the app so it works with different screen resolutions and orientations. At any moment, the horizontal resolution of your app could be dramatically changed. I discuss how to handle switching between different resolutions in Chapter 11, “App Events and States.”

**People, Not Machines, Use Windows Store Apps**

When you buy a Windows Store app, the app is licensed per user and not per machine. When you buy an app, you can use the app on up to five machines—including both tablets and desktops—associated with your user account. You can view and install all of your purchased apps from the Windows Store by right-clicking within the Store app and selecting Your Apps.

Better yet, data from your apps can be shared across multiple machines (roaming application data). So, if you are using an app to read an article on your tablet PC on the bus and then you open the same app on your desktop PC at work, you won’t lose your place in the article.
Creating Your First Windows Store App

Currently, every Windows Store app gets 100KB of roaming application data. Windows 8.1 handles synchronizing this data between different machines for you automatically.

**Closing a Windows Store App**
Now close a Windows Store app by moving your cursor over the x at the top-right of the screen. Ha! Tricked you! There is no close button in a Windows Store app because there is no chrome.

**NOTE**
Even though it is not obvious how to close a Windows Store app, it is possible. You can close a Windows Store app by swiping down from the top of the screen to the very bottom of the screen or pressing the keyboard combination Alt+F4.

When interacting with Windows Store apps, there is no obvious way to close an app. This is intentional. Instead of closing a Windows Store app, you are encouraged to simply switch to another running app (by swiping from the left edge of the screen) or launch a new app (by selecting a new app from the Start screen).

When you design a Windows Store app, you must design the app with the knowledge that a user might switch back and forth to your running app at any time. In Chapter 11 I discuss how you can gracefully resume an app after it has been suspended.

**Creating Your First Windows Store App**
Let’s be fearless. In this section, I guide you through building your first Windows Store app. Doing a *Hello World* app would be predictable and boring. Therefore, I suggest that we do something a little more advanced.

I’ll show you how you can create an app which enables you to take pictures. When you click the Take Picture command in the app bar, you can take a picture, and then the picture is displayed in the app (see Figure 1.6, which shows a picture of my dog Rover).

**NOTE**
The code for the completed app can be found in the Chapter 1 folder with the name App1. All of the code for this book is located in a GitHub repository at https://github.com/StephenWalther/Windows8.1AppsUnleashed.
Creating the Visual Studio Project

The first step is to create a Microsoft Visual Studio Project. I used Visual Studio 2013 to create almost all of the code samples for this book. In most cases, I used the free version of Visual Studio—Visual Studio Express 2013 for Windows—which you can download from Microsoft.com.

NOTE

You can create Windows Store apps with either Microsoft Visual Studio 2013 or Microsoft Blend. If you need to release to the Windows Store then I recommend using Microsoft Visual Studio 2013.

In order to build Windows Store apps, you must use Visual Studio on Windows 8.1. If you don’t have a dedicated Windows 8.1 computer, you can use a virtual machine running Windows 8.1 such as VMware Player.

Go ahead and launch Visual Studio. Next, select the menu option File, New Project. On the left-side of the New Project dialog, select JavaScript and select the Blank App project template. Enter the name App1 for your project and click the OK button (see Figure 1.7).
After you create your project, you can see all of the files for your project in the Solution Explorer window (Figure 1.8). When you create a new Windows Store app, you get a default.html file (in the root of your project), a default.js file (in the js folder), and a default.css file (in the css folder). These three files are the starting point for your app.

Declaring App Capabilities
Before we can jump into writing code, there is one other thing that we must do first. We are building an app that takes pictures. That is scary. Potentially, an app could take pictures of you without your knowledge and send the pictures back to an evil hacker lurking on the Internet (or the CEO of Microsoft).
When your app does something scary, you must declare that your app will do this scary thing up front so the user can consent. You declare the capabilities of your app in your application manifest file. You can open the editor for your application manifest by double-clicking the package.appxmanifest file in the Solution Explorer window.

Click the Capabilities tab to view all of the declared capabilities of your application. For example, if you want your app to be able to record from the computer microphone then you need to select the Microphone capability, or if you want your app to be able to save new photos in the user’s Pictures library then you need to select the Pictures Library capability. For our app, we need to enable the Webcam capability so we can take pictures (see Figure 1.9).

![Figure 1.9 Enabling the capability to take pictures](image)

When a user first runs our app, the user will need to consent to allowing the app to access the webcam (see Figure 1.10). The user only needs to consent once.

![Figure 1.10 Asking for consent to access your webcam](image)
NOTE
After a user consents, the user can deny an app permission to use a particular capability by using the Permissions setting under the Settings charm.

Creating the HTML Page
When you create a Windows Store app, you get a default.html file in the root of your application. This is the first page that is opened when you run your app. Let’s go ahead and customize this page for our picture app (see Listing 1.1).

**LISTING 1.1**  Modified default.html Page

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>App1</title>

<!-- WinJS references -->
<link href="//Microsoft.WinJS.2.0/css/ui-dark.css" rel="stylesheet" />
<script src="//Microsoft.WinJS.2.0/js/base.js"></script>
<script src="//Microsoft.WinJS.2.0/js/ui.js"></script>

<!-- App1 references -->
<link href="/css/default.css" rel="stylesheet" />
<script src="/js/default.js"></script>
</head>
<body>

<img id="imgPhoto" src="/images/placeholder.png" />

<!-- AppBar Control -->
<div id="appBar1" data-win-control="WinJS.UI.AppBar">
  <button data-win-control="WinJS.UI.AppBarCommand"
    data-win-options="{
      id: 'cmdTakePicture',
      label: 'Take Picture',
      icon: 'camera',
      tooltip: 'Take Picture'
    }">
  </button>
</div>
</body>
</html>
```
The HTML page in Listing 1.1 has been modified so it contains new content in the body of the page. First, notice that the page contains an IMG tag with the ID imgPhoto. We’ll display the photo which we take from the camera here.

Notice, furthermore, that the page contains a DIV tag with a data-win-control=“WinJS.UI.AppBar” attribute. This is an example of a WinJS control. This control renders an app bar that contains a command for taking a picture (see Figure 1.11).

![Image of app bar with Take Picture command](image)

**FIGURE 1.11** The Take Picture command in the app bar

### Creating the Style Sheet

When you create a new Windows Store app, you also get a default style sheet named default.css which is located in the css folder. You can modify this file to control the appearance of your app.

For our app, I’ve modified the default.css to format the appearance of the photo. It appears in the IMG tag like this:

```css
#imgPhoto {
    display:block;
    margin: 15px auto;
    border: 10px solid white;
    max-width: 90%;
    max-height: 90%;
}
```

### Creating the JavaScript File

The third file that we need to modify is the JavaScript file named default.js which is located in the js folder. This file contains all of the code associated with the default.html page.

We are going to delete all of the default content of this file and start over. The complete contents of the modified version of default.js are contained in Listing 1.2.
LISTING 1.2  The default.js JavaScript file

```javascript
(function () {

    "use strict";

    // Aliases
    var capture = Windows.Media.Capture;

    // Executed immediately after page content is loaded
    function init() {
        // Process all of the controls
        WinJS.UI.processAll().done(function () {
            // References to DOM elements
            var cmdTakePicture = document.getElementById("cmdTakePicture");
            var imgPhoto = document.getElementById("imgPhoto");

            // Handle Take Picture command click
            cmdTakePicture.addEventListener("click", function () {
                var captureUI = new capture.CameraCaptureUI();
                captureUI.photoSettings.format = capture.CameraCaptureUIPhotoFormat.png;

                captureUI.captureFileAsync(capture.CameraCaptureUIMode.photo).done(function (photo) {
                    if (photo) {
                        // Use HTML5 File API to create object URL to refer to the photo file
                        var photoUrl = URL.createObjectURL(photo);
                        // Show photo in IMG element
                        imgPhoto.src = photoUrl;
                    }
                });
            });
        });
    }

    document.addEventListener("DOMContentLoaded", init);
})();
```

NOTE

The JavaScript code contained in the Default.js file, which we deleted, is used to handle app lifecycle events such as app activation and suspension. I discuss these app events in detail in Chapter 11.
There is a lot of interesting stuff happening in the JavaScript code in Listing 1.2. Let’s walk through the code.

First, I’ve created an `init()` function that is executed when the `DOMContentLoaded` event is raised. The `DOMContentLoaded` event is a standard DOM event that is raised when a browser finishes parsing an HTML document.

I put all of my code into the `init()` function so the code won’t be executed until the DOM is ready. Otherwise, if I attempted to access any of the HTML elements in the page then I would get an exception because the elements would not yet exist.

The first thing that I do within the `init()` method is call the `WinJS.UI.processAll()` method. This method processes all of the controls in a page. In particular, it converts the DIV tag with the `data-win-control="WinJS.UI.AppBar"` attribute into an actual app bar.

Next, I setup an event handler for the Take Picture command. When you click the Take Picture command in the app bar, an instance of the `Windows.Media.Capture.CameraCaptureUI` class is created. The `CameraCaptureUI` class is an example of a Windows Runtime class.

The `CameraCaptureUI.captureFileAsync()` method displays the screen for taking a picture (see Figure 1.12). When you click the OK button, the `done()` method is called and the picture is displayed in the page.

An object URL is created for the photo blob (the actual image data) returned by the `captureFileAsync()` method by calling the `URL.createObjectURL()` method. This `createObjectURL()` method is part of the HTML5 File API.

![Figure 1.12 The camera capture UI screen](image-url)
The photo is displayed in the HTML page with the following line of code:

```javascript
// Show photo in IMG element
imgPhoto.src = photoUrl;
```

And that is all there is to it! We built an app that enables us to take pictures from our computer and display the pictures in an HTML page.

Notice that our JavaScript file contains a combination of standard JavaScript methods, HTML5 methods, Windows Library for JavaScript methods, and Windows Runtime methods. This is normal for all of the JavaScript files that you create when creating a Windows Store app.

**Running the App**

After you create the app, you can run it by pressing the green Run button in the Visual Studio toolbar (see Figure 1.13) or just press the F5 key.

![Running a Windows Store app](image)

**WARNING**

Remember that the Take Picture command is contained in the app bar and the app bar does not appear by default. You need to either right-click the app or swipe from the top or bottom edge of your computer to display the app bar.

**Elements of a Windows Store App**

As we saw in the previous section, a Windows Store app is built using several technologies. A Windows Store app is built out of a combination of open and familiar web technologies, such as HTML5, JavaScript, and CSS3 and Microsoft technologies such as the Windows Library for JavaScript and the Windows Runtime. Let me say a little more about each of these elements of a Windows Store app.

**JavaScript**

This book is all about writing Windows Store apps using JavaScript. As an alternative to JavaScript, you also could write Windows Store apps using C#, Visual Basic, or even C++.

When writing Windows Store apps, you can take advantage of the features of ECMAScript 5 which is the latest version of JavaScript. This means that you can use the new JavaScript
Array methods such as `indexOf()` and `forEach()`. You also can use property setters and getters and the `use strict` statement.

**HTML5**

When writing Windows Store apps, you can take advantage of many of the new features of HTML5 and related standards. Here is a list of some of the most important of these new features:

- **Form Validation Attributes**—You can take advantage of the new validation attributes in the HTML5 standard to perform form validation. I discuss these new validation attributes and how you can use them in a Windows Store app in Chapter 5, “Creating Forms.”
- **data-***—The `data dash star` standard enables you to add custom attributes to existing HTML5 elements. The WinJS library uses `data-*` for declarative data-binding and declarative control instantiation.
- **Indexed Database API (IndexedDB)**—The Indexed Database API exposes a database in the browser. If you need to store a list of products in a database within a Windows Store app, then you can take advantage of IndexedDB. I explain how to use IndexedDB in Chapter 9, “Creating Data Sources.”
- **File API**—The HTML5 File API enables you work with files in the browser. We used the HTML5 API in the previous section when building our first Windows Store app (the `URL.createObjectURL()` method).
- **Canvas**—Enables you to draw graphics using JavaScript. I provide you with an introduction to Canvas in Chapter 15, “Graphics and Games.”
- **Web Workers**—Enables you to execute background tasks without blocking the user interface thread.
- **WebGL**—This is new with Windows 8.1. WebGL enables you to build 3D games with JavaScript.

**Cascading Style Sheets 3**

When you build Windows Store apps, you can take advantage of several new features of the Cascading Style Sheets 3 standard (and related standards) including the following:

- **Media Queries**—Enables you to apply different styles depending on the characteristics of a device, such as the height, width, or orientation of the device. I discuss Media Queries in Chapter 11.
- **CSS3 Grid Layout**—Enables you to lay out HTML content in columns and rows without using HTML tables.
- **CSS3 Flexible Box Layout (FlexBox)**—Enables you to preserve relative element position and size when displaying HTML content in different devices.
**Windows Runtime**

The Windows Runtime (WinRT) contains a class library that you can use in your Windows Store apps. These classes are projected directly into JavaScript, so they appear to be built-in JavaScript objects.

For example, when we wrote our first Windows Store app, we took advantage of the WinRT `Windows.Media.Capture.CameraCaptureUI` class. When we called the `CameraCaptureUI.captureFileAsync()` method, we were able to take a picture.

All of the WinRT classes are exposed in JavaScript from the root Windows namespace. For example, you create an instance of the `CameraCaptureUI` class with the following code:

```javascript
var captureUI = new Windows.Media.Capture.CameraCaptureUI();
```

**NOTE**

Notice that WinRT class names can get silly long. For this reason, it is a good idea to alias the namespaces like this:

```javascript
var capture = Windows.Media.Capture;
```

The WinRT classes extend JavaScript with all of the functionality that you need when building a Windows application. These classes enable you to do fun and amazing things such as:


- **File Access**—Read and write to the file system by taking advantage of the WinRT classes in the `Windows.Storage` namespace.

- **Compass**—Always know the direction of True North with the `Windows.Devices.Sensors.Compass` class.

- **Print**—Print from your Windows Store app by using the `Windows.Printing.PrintManager` class.

- **Compress Files**—Compress and decompress files using the classes in the WinRT `Windows.Storage.Compression` namespace.

**Windows Library for JavaScript**

The Windows Library for JavaScript (WinJS) is a pure JavaScript library created by Microsoft specifically for building Windows Store apps. Understanding how to use this library is the primary focus of this book.
The WinJS library contains all of the WinJS controls. These are the controls that you use to build the user interface for your Windows Store app. For example, the WinJS library includes a DatePicker control that displays a user interface widget for selecting a date.

**What About jQuery?**

jQuery is the most popular JavaScript library in the universe. An obvious question, therefore, is can you use jQuery when building Windows store apps?

---

**NOTE**

According to BuiltWith, more than 57% of the top 10,000 websites use jQuery. This is (by a wide margin) the most common JavaScript framework used on websites. See [http://trends.BuiltWith.com/javascript](http://trends.BuiltWith.com/javascript).

---

The answer is yes. You can use jQuery when building Windows Store apps. Let me show you.

The easiest way to add jQuery to a Windows Store app project is to use the Library Package Manager in Visual Studio. Select the menu option Tools, Library Package Manager, Package Manager Console. Enter the command `install-package jQuery` into the Package Manager Console window (see Figure 1.14).

![Figure 1.14 Adding jQuery with the Library Package Manager Console](image)

Executing the `install-package jQuery` command adds a Scripts folder with four files: the full version of jQuery, the minified version of jQuery, an IntelliSense file, and a source map. The IntelliSense file enables Visual Studio to provide jQuery intellisense when you use jQuery methods and the source map provides debugging support.

Listing 1.3 contains a combined HTML and JavaScript file that uses jQuery.
The page in Listing 1.3 contains a Button and a DIV element. The contents of the DIV element are hidden by default (with `display:none`). When you click the button, the contents of the DIV `fade` slowly into view (see Figure 1.15).
Microsoft worked directly with the jQuery team to ensure that jQuery 2.0 works correctly with Windows Store apps. As long as you are using a version of jQuery more recent than jQuery 2.0 then you should not encounter any issues.

**WARNING**

The fact that Windows Store apps are compatible with jQuery does not mean that Windows Store apps are compatible with every jQuery plugin or popular JavaScript library. In a Windows Store app, JavaScript code executed in the local context has extra security restrictions to prevent script injection attacks. In particular, you cannot assign HTML to the `innerHTML` property, which contains potentially dangerous content such as scripts or malformed HTML.

If you are using a JavaScript library that was not written with these security restrictions in mind then you will need to modify the library to work with a Windows Store app. If you trust the content being assigned to the `innerHTML` property then instead of using the `innerHTML` property, you can use the `WinJS.Utilities.setInnerHTMLUnsafe()` method.

**Building Windows Store Apps with Visual Studio**

This book focuses on building Windows Store apps with Visual Studio. In this section, I want to devote a few pages to describing the features of Visual Studio that matter when building Windows Store apps. You learn how to select a project template for a Windows Store app, how to run a Windows Store app, and how to debug a Windows Store app.
Windows Store App Project Templates
When you select the File, New Project menu option in Visual Studio, you can select from five different project templates as your starting point for your Windows Store app:

2. Navigation App—Use this template for apps that require multiple pages.
3. Grid App—Contains three pages for displaying groups of items.
4. Split App—Contains two pages for displaying groups of items.
5. Hub App—Contains three pages. One page displays a navigation hub and the other two pages display section and item detail. This project template is new with Windows 8.1.

We already used the Blank App project template when creating our first Windows app. Let me discuss the other project templates in more detail.

Navigation App Project Template
The Blank App template is a good template to use when building a simple, single-page app. If you need to support multiple pages, on the other hand, then you should use the Navigation App template.

The Navigation App project template includes a single page named home. You can add additional pages by adding new Page Controls to the pages subfolder (see Figure 1.16).

I describe how you can create multi-page applications in detail in Chapter 12, “Page Fragments and Navigation.”

FIGURE 1.16 Creating a multi-page app with the Navigation App project template

The next three project templates—the Grid App, the Split App, and the Hub App project templates—are built on top of the Navigation App template. In other words, these project templates are multi-page apps with additional pages.
Grid App Project Template
The Grid App project template contains three pages. The main page displays groups of items in a horizontal scrolling grid. You can click a group to view group details or click an item to view item details.

Imagine, for example, that you are creating a product catalog. In that case, you might create different product categories such as Beverages and Fruit. Each category is a group and each category contains a set of product items.

You can use the Grid App groupedItems page to display a horizontal scrolling grid of the product categories and associated products (see Figure 1.17). If you click a product category then you can view details for that category (see Figure 1.18). If you click a product then you can view details for that product (see Figure 1.19).

FIGURE 1.17 The Grid App groupedItems page
Split App Project Template

The Split App project template also can be used to display groups of items such as products grouped into product categories. The Split App project template has two pages: items and split.

The items page displays the list of groups. For example, in Figure 1.20, the items page displays the product categories.
FIGURE 1.20 The Split App items page

If you click a group then you navigate to the split page. This page displays a list of items in the group—the products in the category—and enables you to select an item to see item details (see Figure 1.21).

FIGURE 1.21 The Split App split page

Hub App Project Template

The Hub App project template is new with Windows 8.1. The Hub App template consists of three pages. The main page contains a Hub control and displays a horizontal list of sections (see Figure 1.22). If you click a section title then you navigate to the section page. If you click an item then you navigate to the item page.

NOTE

The Hub control is covered in Chapter 4, “Using WinJS Controls.”
The special thing about the Hub App template is that you can display anything you want within the Hub sections. You can display a list of items, you can display a paragraph of text, or you can display anything else which you heart desires.

For example, in Figure 1.22, Section 1 contains a paragraph of text and Section 2 contains a list of items. Each Hub section can contain different types of content.

![HubApp1](image)

**FIGURE 1.22** The Hub App template

### Running a Windows Store App

Visual Studio provides you with three different options for running a Windows Store app:

- **Local Machine**
- **Simulator**
- **Remote Machine**

The Local Machine option runs a Windows Store app as if the app was installed on the local machine. The Windows Store app will run using the screen resolution and capabilities of your development machine (the machine running Visual Studio).

The Simulator option runs your app in a separate window (see Figure 1.23). The advantage of using the simulator is that you can simulate different types of devices. For example, you can switch from *mouse mode* to *basic touch mode* to simulate a touch device such as a tablet PC. You also can switch to different screen resolutions to test your app at different resolutions.

The final option is to deploy and run your Windows Store app on a remote machine. Before you can run your app on a remote machine, you must first specify the remote machine name in the Project Property Pages window (see Figure 1.24).
CHAPTER 1 Building Windows Store Apps

FIGURE 1.23 Using the Visual Studio simulator

FIGURE 1.24 Specifying the remote machine name
After you specify the name of the remote machine, you can deploy and run your app on the remote machine by picking this option from the Visual Studio toolbar.

**WARNING**

To deploy and run an app on a remote machine, you need to install the Remote Tools for Visual Studio 2013 on the remote machine. You can download the Remote Tools from the Microsoft.com website.

---

**Debugging a Windows Store App**

I’m always optimistic and believe that any code that I write will run without error the first time that I run it. To date, that has never happened. I spend a significant amount of my time debugging code that does not do what I want it to do.

In this section, I discuss the tools in Visual Studio that you can use to debug your code. I discuss how you can use the JavaScript Console window, use breakpoints, and use the DOM Explorer.

**Using the Visual Studio JavaScript Console Window**

When I write JavaScript code for pages used in websites, I use the JavaScript console window to view JavaScript errors. I also write custom messages to the console window using `console.log()` so I can debug my code. (See Figure 1.25.)

![Figure 1.25 Debugging with the Google Chrome JavaScript console](image)

When running a Windows Store app, you don’t have access to the browser JavaScript console. Instead of using the browser JavaScript console, you need to use the Visual Studio JavaScript Console (see Figure 1.26).
You can view JavaScript errors and write debug messages to the Visual Studio JavaScript console window by using `console.log()` in exactly the same way as you would write to a browser console window.

If you hit an error and you want to display the value of a JavaScript variable then you can enter the variable name in the bottom of the JavaScript Console (see Figure 1.27).

**NOTE**

The Visual Studio Console window only appears when an app is running. If you can’t find the window, use the menu option Debug, Windows, JavaScript Console.

**Setting Breakpoints**

If you are building a Windows Store app, and the Windows Store app is behaving in ways that you don’t understand, then it is useful to set breakpoints and step through your code.

You set a breakpoint by clicking in the left gutter of the Visual Studio code editor next to the line that you want to break on (see Figure 1.28). When you run your app in debug mode, and the breakpoint is hit, you can examine the values of your variables by hovering over them with a mouse.
You can step through your code, line by line, by using the Step Into toolbar button or by pressing F11.

**NOTE**
As an alternative to setting a breakpoint with Visual Studio, you can create a breakpoint in code by using the JavaScript `debugger` statement.

**Using the DOM Explorer**
Another of my favorite browser developer tools is the HTML inspector (this is a feature, for example, of Firebug). You can use this tool to view the live HTML and CSS in a document.

Visual Studio supports a similar tool named the DOM Explorer. You can use the DOM Explorer to inspect the property of any HTML element in a running Windows Store app.

After running a Windows store app in Visual Studio, you can view the DOM Explorer window by selecting the menu option Debug, Windows, DOM Explorer. Within the DOM Explorer window, you can click any element and view all of the properties of the element including information about all of the styles associated with the element (see Figure 1.29).
If you click an element associated with a WinJS control then you can see all of the HTML attributes and elements rendered by the control. Adding a ListView control to a page, for example, adds a lot of new DIV elements to the page.

**Publishing to the Windows Store**

One of the main motivations for building a Windows Store app is to sell your app in the Windows Store for either fame or profit. In this section, I discuss the steps you need to follow to publish your Windows Store app to the Windows Store.

**NOTE**


**Register as a Windows Developer**

Before you can publish an app to the Windows Store, you must first register as a Windows Store developer. You can sign up at the Windows Store Dashboard on the Windows Dev Center by selecting the menu option Project, Store, Open Developer Account within Visual Studio (see Figure 1.30).
The sign-up procedure is painless. Currently, it costs either $49 (for an individual account) or $99 (for a company account) a year to become a registered Windows Store developer, or it is free with a MSDN subscription.

Submitting Your App
After you register, you can access the Windows Store dashboard and submit a new app. The process of submitting an app is broken down into 8 steps (see Figure 1.31).

One of the most important steps is selecting the name for your app. You can reserve an app name in the Windows Store even before you have finished creating the app. Picking an app name is similar to picking a domain name—so I recommend that you acquire the name that you want as soon as possible.

You also need to decide on how much you want to charge for your app. Currently, you can charge anywhere from $1.49 to $999.99. Or, you have the option of providing your app for free. You also have the option of providing your app with a limited free trial or making your app free with advertising.
NOTE

There are iPhone apps that sell for $999.99 dollars. For example, the iVIP Black iPhone app sells for $999.99. But to purchase it, you need to prove that you are a “High Net Worth” individual with “assets and/or income in excess of £1 million.”

FIGURE 1.31  Submitting an app to the Windows Store

When you reach the sixth step, the Packages step, you can upload your finished Windows Store app to the Windows Store. Within Visual Studio, use the menu option Project, Store, Create App Package to package up your Windows Store app (see Figure 1.32). Next, you can click the Packages step to upload the package.
Passing App Certification

Microsoft must review your app before it gets published to the Windows Store. In other words, your app must go through a certification process. Part of this certification process is automated and part of the certification process must be done by a human.

There are many requirements for certification. Some of these requirements are obvious. For example, your app can’t contain programming errors that cause it to immediately crash and your app cannot simply be a big ad for your business.

Some of the certification requirements are not so obvious. For example, to be certified, your app cannot unexpectedly transport large amounts of data over a metered network connection, your app must start up quickly, and your app must be complete (no “coming soon” features). Also, if your app links to the Internet, you must provide a privacy policy.

NOTE

You can use the Windows App Certification Kit to run the automated certification tests on your app before you upload your package to the Windows Store. The easiest way to run the Windows Certification Kit is to package your app within Visual Studio by selecting the menu option Package, Store, Create App Package. The last step in the Create App Package Wizard enables you to launch the Windows App Certification Kit (Figure 1.33).
NOTE

The Windows App Certification Kit is installed at the same time as you install Visual Studio. You can run it independently of Visual Studio by launching the Windows App Cert Kit from the Start screen.

When you run the Windows App Certification Kit, the App Certification Kit launches and runs your app and then, after your computer does crazy stuff for a while, a report is generated that details whether your app passes or fails (see Figure 1.34).

NOTE

If you are using Team Foundation Server, you can even integrate the Windows App Certification Kit into your build process. Every time you do a new build of your app, you can run the technical certification tests automatically.
Migrating from Windows 8 to Windows 8.1

Windows 8.1 is the second release of Windows 8. There are already tens of thousands of apps written for Windows 8.

If you already created a Windows Store app for Windows 8 and you want to migrate the app to Windows 8.1 then the process is dead easy. When you open your Windows 8 app in Visual Studio 2013, Visual Studio recommends retargeting your app to Windows 8.1 (see Figure 1.35).

FIGURE 1.34 A (successful) certification report generated by the Windows App Certification Kit

After your app passes all the certification requirements—after it has been approved by Microsoft—your app appears in the Windows Store and you can start collecting money. When anyone buys your app, money is added to a payout account, which you set up on the Windows Store dashboard.

Migrating from Windows 8 to Windows 8.1

Windows 8.1 is the second release of Windows 8. There are already tens of thousands of apps written for Windows 8.

If you already created a Windows Store app for Windows 8 and you want to migrate the app to Windows 8.1 then the process is dead easy. When you open your Windows 8 app in Visual Studio 2013, Visual Studio recommends retargeting your app to Windows 8.1 (see Figure 1.35).
You can right-click your project in the Solution Explorer window and select the menu option Retarget to Windows 8.1 to migrate your app to Windows 8.1.

Retargeting your app updates all of your script references to point to the Windows Library for JavaScript 2.0 instead of the Windows Library for JavaScript 1.0. If you prefer, you could do this by hand by adding a reference to the Windows Library for JavaScript 2.0 to your project and updating the `<script>` tags in all of your HTML pages.

After you retarget your app, you might need to make code changes. For example, as I mentioned earlier in this chapter, Windows 8.1, unlike Windows 8, no longer supports a snapped view state. A list of all of the deprecated Windows 8 application programming interface (APIs) is displayed after you retarget your app.

**NOTE**

You need Visual Studio 2013 Professional, Premium, or Ultimate to edit an existing Windows 8 app. Visual Studio 2013 Express requires you to retarget a Windows 8 app to Windows 8.1 before you can modify it.

This might be obvious, but I am going to say it anyway. Apps written for Windows 8.1 won’t run on Windows 8. The Windows Runtime in Windows 8.1 has changed so you won’t see Windows 8.1 apps in the Windows Store on a computer running Windows 8. You still can use Windows 8 apps, on the other hand, with Windows 8.1—you can install both Windows 8 and Windows 8.1 apps from the Windows Store on a computer running Windows 8.1.

**Summary**

The goal of this chapter was to introduce you to Windows Store apps. I started this chapter by providing you with an overview of the Microsoft design style principles. You also learned about the standard features of Windows Store apps such as the app bar and charms.
I then led you, step by step, through the process of building your first Windows Store app. We created a really cool camera app that you could never create as a standard web application.

You also learned about the standard elements of a Windows Store app. You learned how a Windows Store app is composed of standard HTML5, JavaScript, and CSS3. You also learned how Windows Store apps take advantage of Microsoft technologies such as the Windows Runtime and the Windows Library for JavaScript.

I also explained how you can take advantage of the features of Visual Studio when building a Windows Store app. You learned how to run a Windows Store app using the simulator. You also learned how to debug a Windows Store app by using breakpoints and the Visual Studio JavaScript Console window.

Finally, you learned how you can make money from your Windows Store app by publishing your app to the Windows Store. You learned how to register your app, submit your app, and pass certification.
This page intentionally left blank
Symbols
@-ms-viewport rules, 326, 329

A
About Page settings, creating, 187-189
account settings, creating, 396-401
activated events, app events, 312-325
adapting to screen changes (MyTasks app), 440-443
addEventListener(), 315
adding Page Controls to Navigation App, 343-345
search results pages, 373-376
SearchBox control to pages, 377-378
Ajax calls, performing with xhr() function, 69-74
response types, specifying, 72-73
alert.css file, 338
alert.js file, 338
any() method, 63
app bar, 8-9
app events, 311-312
activated events, 312-325
creating custom, 315
deferring events, 314-315
designing apps for different window sizes
CSS media queries, 321-324
ero errors, 313-314
logging, 312
AppBar control, 176-184
commands, 178-181
contextual commands, 181-184
application keys, retrieving Mobile Services, 297
application state, testing with Visual Studio, 317-318
ApplicationExecutionState, 317
applying templates with query selectors, 109-111
apps, 315-316
declaring as share targets, 361-362
designing for different window sizes, 320-325

Index
detecting previous execution state, 316-317
detecting suspended and terminated apps, 316
multi-page apps. See multi-page apps
scaling to fit screen resolutions
    defining viewports, 326-329
    ViewBox control, 329-332
storing state with session state, 318-320
testing application state with Visual Studio, 317-318
articles, displaying with FlipView control, 215-218
assigning format strings to DatePicker control, 128-130
asynchronous programming, promises, 56-63
canceling, 62-63
composing, 63
creating, 60-61
timeout promises, 61-62
audio (Brain Eaters game), playing, 420-421
authentication, 394-401
account settings, creating, 396-401
logging in users to Live Connect, 394-396
Azure Mobile Services. See also Mobile Services
    configuring, 401
    logging into, 404-406
mobile server scripts, updating, 402-404
permissions, setting, 402

B
back(), 346
beforenavigate, Navigation API, 347
binding converters, 101-105
    creating, 103
date and price converters, creating, 104-105
bindings
    declarative data binding, 81, 91-105
    binding converters, 101-105
data context, 94
data-win-bind attribute, 92
HTML forms, capturing contents of, 96-98
observables, 94-96
root element, 94
and WinJS controls, 99-100
object properties, binding to a listener, 83-85
WinJS.Binding.List object, 88-90, 224-226
blog feeds, binding ListView control to, 226-228

Brain Eaters, 417-418
Canvas, creating, 421-423
overview, 418-419
render loop, creating, 427-429
sounds, playing, 420-421
tiles, creating, 419-420
update loop, creating, 425-427
user interaction, capturing, 424-425
breakpoints, setting, 34-35
buttons, creating FlipView custom buttons, 221-222
bypassing notifications, 87-88

callbacks, 57
promises, 57-58
canceling, 62-63
composing, 63
WinJS.xhr() method, 58
canceling promises, 62-63
canGoBack, 347
canGoForward, 347
Canvas, 22
Brain Eaters game, creating, 421-423
capturing
contents of HTML forms, 96-98
selected date with DatePicker control, 132-133
user interaction (Brain Eaters game), 424-425
WebView screenshots, 145-146

Cascading Style Sheets. See CSS (Cascading Style Sheets)
cell spanning layout (ListView control), 231-236
certification, Windows Store, 39-40
chaining, 59
c change() method, implementing, 268
chars, 9-11
Settings charm, 186-187

classes, 51-56
creating, 51-52
QueryCollection class, 68-69
Windows RT, 23
win-item, 199
win-itembox, 199
win-itemcontainer, 199
clearing user ratings, 124-125
closing Windows Store apps, 13
coalesc ing notifications, 85-87
collections, observable collections, 90-91
collections, observable collections, 90-91
commands
app bar commands, 178-181
contextual commands, 181-184
SQL TRUNCATE TABLE, 308
common features of Windows Store apps, 7-13
app bar, 8-9
chars, 9-11
nav bar, 9
CommonJS website, 57
composing promises, 63
configuring Azure Mobile Services, 401
connecting Mobile Services to remote database tables, 299
console.log(), 309
containers. See ItemContainer
cumentEditable attribute, 164-165
contentElement property (ToolTip control), 121
ContentIndexer, 381
ContentIndexerQuery, 381
contextual commands, 181-184
controls, 113-120
AppBar control, 176-184
commands, 178-181
contextual commands, 181-184
creating declaratively, 115-117
creating imperatively, 117-118
DatePicker control, 127-133
decl a ring, 128
format strings, assigning, 128-130
selected date, capturing, 132-133
de claring, 113-114
FlipView
custom buttons, 221-222
displaying articles with, 215-218
displaying page numbers with, 219-220
explained, 197
Flyout control, 169-171
Hub control, 137-139
   sections, navigating, 139
ItemContainer
   combining with Repeater control, 214-215
   dragging and dropping, 204-208
   explained, 197
   invoking, 200-202
   selecting, 202-204
   simple example, 197-198
   styling, 198-200
   swipeBehavior property, 202
   tabBehavior property, 202
ListView control, 223
   binding to a blog feed, 226-228
   dragging and dropping, 256-262
   filtering items, 242-244
   grouping items, 245-247
   invoking items, 236-238
   selecting items, 238-241
   sorting items, 241-242
   templates, switching, 253-255
   views, switching with Semantic Zoom, 248-253
   WinJS.Binding.List data source, 224-226
Menu control, 172-174
NavBar control, 184-186
   options, setting, 118-119
Rating control, 124-127
   declaring, 124-125
   events, 125-127
   ratings, customizing, 125
   ratings, submitting, 125-127
   references, adding, 114-115
Repeater
   explained, 197
   external templates, 210-211
   with ItemContainer, 214-215
   nested templates, 211-213
   simple example, 208-210
   retrieving from HTML documents, 119-120
TimePicker control, 133-136
   current time, setting, 134-136
   declaring, 133-134
   time, formatting, 136
ToggleSwitch control, 122-124
   declaring, 122
   state of, determining, 123-124
ToolTip control, 120-121
   contentElement property, 121
   declaring, 120
   styling, 121
WebView control, 139-146
   events, 142
   navigation, handling, 142-144
   screenshots, capturing, 145-146
   web pages, hosting, 140-142
   winControl property, 99-100
CORS (W3C Cross-Origin Resource Sharing) standard, 71
createObjectURL() method, 20
creating
   About Page settings, 187-189
   AppBar control, 176-178
   binding converters, 103
   Brain Eaters game tiles, 419-420
   classes, 51-52
   custom data sources, 263-270
      change() method, implementing, 268
      error handling, 268-269
      getCount() method, implementing, 265
      itemsFromIndex() method, implementing, 265-267
      remove() method, implementing, 267-268
      setNotificationHandler() method, implementing, 269-270
   data adapters, 264-265
   data sources
      file data sources, 270-276
      IndexedDB data sources, 281-293
   indexes, 284-286
   JavaScript file, 18-21
   MyTasks app, 431-432
   observable collection of observables, 90-91
   observables, 82-83
   personal settings, 189-192
   promises, 60-61

How can we make this index more useful? Email us at indexes@sams publishing.com
creating

rich text editor, 164-165
style sheet, 18
templates
declarative templates, 108-109
external templates, 111-112
imperative templates, 105-108
Visual Studio project, 14-15
app capabilities, declaring, 15-17
web service data sources, 276-281
WinJS controls
creating declaratively, 115-117
creating imperatively, 117-118
cross-origin requests, 71
CSS (Cascading Style Sheets)
ItemContainer
styling, 198-200
media queries, 321-324
selectors, 64
CSS3 (Cascading Style Sheets 3), 22
current time, setting with TimePicker control, 134-136
custom actions, performing with Mobile Services, 306-308
custom app events, creating, 315
custom buttons, creating in FlipView, 221-222
custom data sources, creating, 263-270
change() method, implementing, 268
error handling, 268-269
getCount() method, implementing, 265
remove() method, implementing, 267-268
setNotificationHandler() method, implementing, 269-270
customizing
Rating control ratings, 125
validation error style, 152-154

D
data adapters, creating, 264-265
data context, 94
data sources
creating, 263-270
change() method, implementing, 268
getCount() method, implementing, 265
itemsFromIndex() method, implementing, 265-267
setNotificationHandler() method, implementing, 269-270
file data sources, creating, 270-276
IndexedDB data sources, creating, 281-293
web service data sources, creating, 276-281
data+, 22
database data
deleting in Mobile Services, 301
inserting in Mobile Services, 299-300
updating with Mobile Services, 300
database tables, creating in Mobile Services, 297-298
DataPackage, 358
DataPackageView class, 366
datarequested handler, 358
data-win-bind attribute, 92
binding converters, 101-105
date and price converters, creating, 104-105
DatePicker control, 127-133
days, hiding from display, 131-132
declaring, 128
format strings, assigning, 128-130
selected date, capturing, 132-133
years, hiding from display, 131-132
days, omitting from DatePicker control, 131-132
debugging
script errors, Mobile Services, 308-309
Windows Store apps in Visual Studio, 33-36
breakpoints, setting, 34-35
DOM Explorer, 35-36
JavaScript Console window, 33-34
declarative data binding, 81, 91-105
binding converters, 101-105
creating, 103
date and price converters, creating, 104-105
data context, 94
data-win-bind attribute, 92
HTML forms, capturing contents of, 96-98
observables, 94-96
root element, 94
and WinJS controls, 99-100
declarative templates, creating, 108-109
declaring
apps
capabilities, 15-17
as search providers, 369-370
as share targets, 361-362
WinJS controls, 113-114
  DatePicker control, 128
  Rating control, 124-125
  TimePicker control, 133-134
  ToggleSwitch control, 122
  ToolTip control, 120
default.html page, Navigation App, 341-342
default.js file, creating, 18-21
deferring app events, 314-315
del() method, 301
deleting database data (Mobile Services), 301
designing apps for different window sizes, 320
  CSS media queries, 321-324
  setting minimum app width, 320-321
  window resize events, 324-325
detecting
  previous execution state of apps, 316-317
  suspended and terminated apps, 316
determinate progress indicator, displaying, 167
dialogs, displaying, 192-194
displaying
  articles with FlipView control, 215-218
  dialogs, 192-194
  Flyout controls, 169-171
  page numbers with FlipView control, 219-220
  progress indicator, 165-167
DOM element, retrieving WinJS controls from, 119-120
DOM Explorer, 35-36
done() method, 59-60
downloading files from SkyDrive, 411-413
dragend event, 204
dragenter event, 204
dragging and dropping
  ItemContainer, 204-208
  ListView control items, 256-262
dragleave event, 204
dragover event, 204
dragstart event, 204
E
  elements of Windows Store apps, 21-26
    CSS3, 22
    HTML5, 22
    JavaScript, 21-22
    JQuery, 24-26
    Windows Library for JavaScript, 23-24
    Windows RT, 23
e-mail addresses, entering in forms, 160-161
embedding web pages in Windows Store apps, 139-146
encapsulating methods, 49-51
error handlers
  app events, 313-314
  custom data sources, 268-269
  insert() function, 305
events
  app events, deferring, 314-315
  dragend, 204
  dragenter, 204
  dragleave, 204
  dragover, 204
  dragstart, 204
  iteminvoked, 200
  for Ratings control, 125-127
  selectionchanged, 202
  selectionchanging event, 202
external templates
  creating, 111-112
  with Repeater control, 210-211
F
  File API, 22
  file data sources, creating, 270-276
  files, SkyDrive
    downloading from, 411-413
    listing, 409-411
    uploading, 413-415
  filtering ListView control items, 242-244
  FlipView
    custom buttons, 221-222
    displaying articles with, 215-218

How can we make this index more useful? Email us at indexes@samspublishing.com
displaying page numbers with, 219-220 explained, 197
Flyout control, 169-171
SettingsFlyout, 190-192
folders (SkyDrive), listing, 409-411
format strings, assigning to DatePicker control, 128-130
formatting time (TimePicker control), 136
forms
fields, labeling, 157-158
HTML, capturing contents of, 96-98
input elements, 155-164
e-mail addresses, entering, 160-161
files, selecting, 162-164
numbers, entering, 158-159
search terms, entering, 160-161
television numbers, entering, 160-161
URLs, entering, 160-161
values from a list of values, entering, 162
values from a range, entering, 159-160
progress indicator, displaying, 165-167
resetting, 154-155
rich text editor, creating, 164-165
validation attributes
custom validation, performing, 151-152
pattern attribute, 150-151
required attribute, 150
validation error style, customizing, 152-154
forward(), 347
functions, 49-51
init(), 20
xhr(), performing Ajax calls with, 69-74

G
games, Brain Eaters, 417-418
overview, 418-419
tiles, creating, 419-420
getBitmapAsync(), 367
getCount() method, implementing, 265
dataAsync(formatId), 367
dataHtmlFormatAsync(), 367
dataRtfAsync(), 367
dataStorageItemsAsync(), 367
dataTextAsync(), 367
dataWebLinkAsync(), 367
Grid App project template, 28
ggrid layout (ListView control), 229-230
grouping ListView control items, 245-247

H
handling navigation with WebView control, 142-144
hiding years and days from DatePicker control, 131-132
higher priority jobs, yielding to, 77-80
history, 347
hosting web pages with WebView control, 140-142
HTML page, creating, 17-18
HTML5, 22
Canvas, 22
data*, 22
declarative data binding, 91-105
binding converters, 101-105
data context, 94
data-win-bind attribute, 92
HTML forms, capturing contents of, 96-98
observables, 94-96
root element, 94
File API, 22
forms
e-mail addresses, entering, 160-161
files, labeling, 157-158
files, selecting, 162-164
numbers, entering, 158-159
progress indicator, displaying, 165-167
resetting, 154-155
rich text editor, creating, 164-165
search terms, entering, 160-161
telephone numbers, entering, 160-161
URLs, entering, 160-161
values from a list of values, entering, 162
values from a range, entering, 159-160
Indexed Database API, 22
input elements, 155-164
templates
applying with query selector, 109-111
declarative templates, 108-109
ListView control

imperative templates, creating, 105-108
imperative templates, performing, 151-152
validation attributes, 22
custom validation, 150-152
required attribute, 150
validation error style, customizing, 152-154
Web Workers, 22
WebGL, 22
HtmlControl, 333-336
Hub App project template, 28-30
Hub control, 137-139
sections, navigating, 139

labels
imperative templates, creating, 105-108
indeterminate progress indicator, displaying, 165-166
IndexableContent, 381
Indexed Database API, 22
IndexedDB data sources, creating, 281-293
Indexer Helper, 382-383
Indexer helper object, creating, 381-382
Indexer query() method, 384
indexes, creating, 284-286
init() function, 20
initializing Live Connect API connection, 391
input elements, 155-164
email addresses, entering, 160-161
files, selecting, 162-164
numbers, entering, 158-159
search terms, entering, 160-161
telephone numbers, entering, 160-161
URLs, entering, 160-161
values from a list of values, entering, 162
values from a range, entering, 159-160
insert() function, 305
error handlers, 305
insert() method, 300
inserting database data, Mobile Services, 299-300
installing
Live SDK, 388-393
Mobile Services for WinJS library, 298-299
invokeApi(), 308
invoking
ItemContainer, 200-202
ListView control items, 236-238
ItemContainer
combining with Repeater control, 214-215
dragging and dropping, 204-208
explained, 197
invoking, 200-202
selecting, 202-204
simple example, 197-198
styling, 198-200
swipeBehavior property, 202
tabBehavior property, 202
iteminvoked event, 200
itemsFromIndex() method, implementing, 265-267
itemTemplate property, FlipView control, 218

J-K
JavaScript, 21-22
JavaScript file, creating, 18-21
jobs, prioritizing with Scheduler, 75-80
join() method, 63
JQuery, 24-26

labeling form fields, 157-158
launching Windows App Certification kit, 40-41
layouts for ListView control
  cell spanning layout, 231-236
  grid layout, 229-230
  list layout, 231
light dismiss, 169
list layout (ListView control), 231
listeners
  binding object properties to, 83-85
  notifications, coalescing, 85-87
  observables, 81-91
  creating, 82-85
  registering, 83
listing SkyDrive files and folders, 409-411
ListView control, 223, 349

How can we make this index more useful? Email us at indexes@samspublishing.com
ListView control

binding to a blog feed, 226-228
dragging and dropping, 256-262
filtering items, 242-244
grouping items, 245-247
invoking items, 238-236
layouts
   cell spanning layout, 231-236
   grid layout, 229-230
   list layout, 231
reordering items, 256
selecting items, 238-241
sorting items, 241-242
templates, switching, 253-255
views, switching with Semantic Zoom, 248-253
WinJS.Binding.List data source, 224-226

Live Connect API, 387
   apps, registering, 389-391
   authentication, 394-401
      account settings, creating, 396-401
      logging in users, 394-396
Azure Mobile Services
   configuring, 401
   logging into, 404-406
   mobile server scripts, updating, 402-404
   permissions, setting, 402
   connection, initializing, 391
   scopes, specifying, 391-393
   user information, retrieving, 406-408
Live SDK
   installing, 388-393
   references, adding, 388
Live Services, 387
location, 347
logging in users to Live Connect, 394-396
logging into Azure Mobile Services, 404-406
logging WinJS app events, 312
looking up a single database record, 301-302

M
make.js, 307
media queries, CSS media queries, 321-324
Menu control, 172-174
methods
   any(), 63
   chaining, 59
   createObjectURL(), 20
   done(), 59-60
   encapsulating, 49-51
   join(), 63
   private methods, 49
   processAll(), 219
   public methods, 48
   in QueryCollection class, 68-69
   then(), 59-60
   WinJS.Class.define(), 51-52
   WinJS.Class.derive(), 53-54
   WinJS.Class.mix(), 54-56
   WinJS.Namespace.define(), 46-48
   WinJS.Namespace.defineWithParent(), 48
   WinJS.UI.processAll(), 20
   WinJS.Utilities.children(), 67-68
   WinJS.Utilities.id(), 66-67
   WinJS.Utilities.query(), 64-66
   WinJS.xhr(), 58
Metro design principles. See Microsoft design style principles
Microsoft design style principles, 6-7
migrating from Windows 8 to Windows 8.1, 40-42
minimum app width, setting, 320-321
mixins, 54-56
Mobile Services, 295
   application keys, retrieving, 297
   configuring, 401
   connecting to remote database tables, 299
   creating, 295-297
   database tables, creating, 297-298
   debugging script errors, 308-309
   deleting database data, 301
   inserting database data, 299-300
   installing for WinJS library, 298-299
   performing
      custom actions, 306-308
      validation, 304-306
   permissions, setting, 402
queries
   looking up a single database record, 301-302
retrieving a set of database records, 302-304
scripts, updating, 402-404
updating database data, 300
MobileServiceClient invokeApi() method, 308
MobileServiceTable object, 302
modal dialogs, displaying, 192-194
modules, 48-51
multi-page apps, 340
  navigating to another page, 345-346
Navigation API, 346-347
Navigation App
  adding Page Controls, 343-345
  creating, 340-341
  default.html page, 341-342
Navigation state, 347-351
PageControlNavigator control, 347
My Notes app, 353
MyTasks app, 432-433
  creating, 431-432
  custom control, creating, 444-446
  external services, connecting to, 435-437
  optimistic inserts, 437-439
  screen changes, adapting to, 440-443
  setting up, 433-434
Text-to-Speech API, 446-448

namespaces, 46-48
  WinJS.Namespace.define() method, 46-48
  WinJS.Namespace.defineWithParent() method, 48
nav bar, 9
NavBar control, 184-186
navigate(), 347
navigating
  hub sections, 139
  Navigation API, 347
  to other pages in multi-page apps, 345-346
navigation, controlling with WebView control, 142-144
Navigation API, 346-347
Navigation App
  adding Page Controls, 343-345
creating, 340-341
default.html page, 341-342
Navigation App project template, 27
Navigation state, multi-page apps, 347-351
nested templates with Repeater control, 211-213
notifications
  bypassing, 87-88
  coalescing, 85-87
numbers, entering in forms, 158-159

0
objects
  observable collections, creating, 90-91
  properties, binding to a listener, 83-85
objects stores
  adding objects to, 283
  determining number of items in, 283-284
observable collections, 90-91
observables, 81-91
  and declarative data binding, 94-96
  listeners
    creating, 83-85
    notifications, bypassing, 87-88
    notifications, coalescing, 85-87
observable collections, 90-91
OData, 303-304
omitting years and days from DatePicker control, 131-132
onShareSubmit() method, 365
optimistic inserts (MyTasks app), 437-439
options, setting for WinJS controls, 118-119
organizing code
  classes, 51-56
  modules, 48-51
  namespaces, 46-48
    WinJS.Namespace.define() method, 46-48
    WinJS.Namespace.defineWithParent() method, 48
orientation
  media queries, 18
  of Windows Store apps, 11-12

How can we make this index more useful? Email us at indexes@samspublishing.com
Page Controls, 333
adding to Navigation App, 343-345
creating, 336-340
page numbers, displaying with FlipView control, 219-220
PageControlNavigator, 333
PageControlNavigator control, 347
pages adding, SearchBox control, 377-378
passing app certification, 39-40
pattern attribute, 150-151
performing
custom actions (Mobile Services), 306-308
validation (Mobile Services), 304-306
permissions, revoking for Windows Store apps, 393
personal settings, creating, 189-192
popups, Flyout control, 169-171
price converter, creating, 104-105
prioritizing jobs with Scheduler, 75-80
private methods, 49
processAll() method, 219
progress indicator, displaying, 165-167
promises, 56-63
canceling, 62-63
chaining, 59
composing, 63
creating, 60-61
done() method, 59-60
then() method, 59-60
timeout promise, creating, 61-62
WinJS.xhr() method, 58
properties
binding to a listener, 83-85
declarative data binding, binding converters, 101-105
observables, 81-91
creating, 82-83
winControl property, 99-100
public methods, 48
publishing to the Windows Store, 36-40
passing app certification, 39-40
registering as Windows Developer, 36
submitting your app, 37-38
queries (Mobile Services)
looking up a single database record, 301-302
retrieving a set of database records, 302-304
query selectors, 63-69
QueryCollection class, 68-69
templates, applying, 109-111
WinJS.Utilities.children() method, 67-68
WinJS.Utilities.id() method, 66-67
WinJS.Utilities.query() method, 64-66
QueryCollection class, 68-69
querying Windows Index, 383-384
Rating control, 124-127
declaring, 124-125
events, 125-127
ratings
customizing, 125
submitting, 125-127
references
adding to controls, 114-115
adding to Live SDK, 388
registering
apps with Live Connect, 389-391
listeners, 83
as Windows Developer, 36
remote database tables, connecting (Mobile Services), 299
remove() method, implementing, 267-268
render loop (Brain Eaters game), creating, 427-429
reordering ListView items, 256
Repeater
explained, 197
external templates, 210-211
with ItemContainer, 214-215
nested templates, 211-213
simple example, 208-210
required attribute, 150
resetting forms, 154-155
resolution, 323
scaling apps to fit
  defining viewports, 326-329
ViewBox control, 329-332
retargeting to Windows 8.1, 40-42
retrieving
  Mobile Service application keys, 297
  sets of database records, 302-304
  user information, 406-408
  WinJS controls from HTML documents, 119-120
revoke Windows Store app permissions, 393
rich text editor, creating, 164-165
root element, 94
running apps, 21

S
SaveNote(), 368
scaling apps to fit screen resolutions
  defining viewports, 326-329
ViewBox control, 329-332
Scheduler, 75-80
scopes, specifying for Windows apps, 391-393
screen changes, adapting to (MyTasks app), 440-443
screen resolution, 323
  of Windows Store apps, 11-12
screenshots (WebView), capturing, 145-146
script errors, debugging with Mobile Services, 308-309
search activation, Search charm, 372-373
Search charm, 368-369
  declaring apps as search providers, 369-370
  search activation, 372-373
  search results pages, adding, 373-376
  search suggestions, 370-372
search results, displaying (SearchBox control), 379
search results pages, adding, 373-376
search suggestions
  Search charm, 370-372
  SearchBox control, 378-379
search terms, entering in forms, 160-161
SearchBox control, 376-377
  adding to pages, 377-378
  search results, displaying, 379
  search suggestions, 378-379
sections (hub), navigating, 139
security, user authentication, 394-401
  account settings, creating, 396-401
  logging in users to Live Connect, 394-396
selecting
  ItemContainer, 202-204
  ListView control items, 238-241
selectionchanged event, 202
selectionchanging event, 202
selectors, 63-69
  QueryCollection class, 68-69
  templates, applying, 109-111
  WinJS.Utilities.children() method, 67-68
  WinJS.Utilities.id() method, 66-67
  WinJS.Utilities.query() method, 64-66
Semantic Zoom, 248-253
server insert.js script, 305
session states, storing with states, 318-320
sessionState/sessionState.js, 318-319
setHtmlFormat(), 359
setNotificationHandler() method, implementing, 269-270
sets of database records, retrieving with Mobile Services, 302-304
setText, 359
setting breakpoints, 34-35
settings
  About Page settings, creating, 187-189
  personal settings, creating, 189-192
Settings charm, 186-187
Settings flyout, 190-192
Share charm, 355-356
Share Contract Target JavaScript file, 364-365
share pages, creating, 362-368
share sources, creating, 356-360
share targets
  creating, 360-368
  declaring apps as, 361-362
sharing, 354-356
  share pages, creating, 362-368
  share sources, creating, 356-360

How can we make this index more useful? Email us at indexes@samspublishing.com
share targets
  creating, 360-368
  declaring apps as, 361-362
  Windows Store apps across devices, 12-13
SkyDrive
  files
    downloading, 411-413
    listing, 409-411
    uploading, 413-415
  folders, listing, 409-411
  sorting ListView control items, 241-242
sounds (Brain Eaters game), playing, 420-421
Split App project template, 29-30
SQL Azure database tables, creating new, 297-298
SQL TRUNCATE TABLE, 308
SSML (Speech Synthesis Markup Language), 448
state
  storing with session state, 318-320
state of ToggleSwitch control, determining, 123-124
storing state with session state, 318-320
style sheets, creating, 18
  ItemContainer, 198-200
  ToolTip control, 121
submitting
  apps to the Windows Store, 37-38
  user ratings, 125-127
suspended apps, detecting, 316
swipeBehavior property (ItemContainer), 202
switching
  ListView templates, 253-255
  views with Semantic Zoom, 248-253
synchronous programming, 57

T

tabBehavior property (ItemContainer), 202
task list app
  creating, 431-432
  custom control, creating, 444-446
  external services, connecting to, 435-437
  optimistic inserts, 437-439
  screen changes, adapting to, 440-443
  setting up, 433-434
  Text-to-Speech API, 446-448
  telephone numbers, entering in forms, 160-161
  templates, 105-112
  applying with query selector, 109-111
  declarative templates, creating, 108-109
  external templates, creating, 111-112
  imperative templates, creating, 105-108
  ListView templates, switching, 253-255
  Repeater templates
    external templates, 210-211
    nested templates, 211-213
  Windows Store app project templates, 27-30
  Grid App project template, 28
  Hub App project template, 28-30
  Navigation App project template, 27
  Split App project template, 29-30
WinJS, 81
terminateApp(), 313
terminated apps, detecting, 316
testing application state with Visual Studio, 317-318
Text-to-Speech API, 446-448
  then() method, 59-60
threads, prioritizing jobs with Scheduler, 75-80
tiles (Brain Eaters game), creating, 419-420
timeout promises, 61-62
TimePicker control, 133-136
  current time, setting, 134-136
  time, formatting, 136
Timing Control for Script-Based Animation standard, 429
ToggleSwitch control, 122-124
  declaring, 122
  state of, determining, 123-124
ToolTip control, 120-121
  contentElement property, 121
  declaring, 120
  styling, 121

U

update loop (Brain Eaters game), creating, 425-427
updating database data (Mobile Services), 300
uploading files to SkyDrive, 413-415
URLs, entering in forms, 160-161
user experience, Microsoft design style principles, 6-7
user information, retrieving, 406-408
user ratings
  clearing, 124-125
  submitting, 125-127

V
validation, performing with Mobile Services, 304-306
validation attributes, 22
  custom validation, performing, 151-152
  error style, customizing, 152-154
  pattern attribute, 150-151
  required attribute, 150
  validation error style, customizing, 152-154
values from a range, entering in forms, 159-160
ViewBox control, scaling apps to fit different resolutions, 329-332
viewports, defining, 326-329
views, FlipView
  custom buttons, 221-222
  displaying articles with, 215-218
  displaying page numbers with, 219-220
  explained, 197
Visual Studio
  breakpoints, setting, 34-35
  DOM Explorer, 35-36
  JavaScript Console window, 33-34
projects
  app capabilities, declaring, 15-17
  apps, running, 21
  creating, 14-15
  HTML page, creating, 17-18
  JavaScript file, creating, 18-21
  JQuery, adding, 24-26
  style sheet, creating, 18
testing application states, 317-318
Windows Store app project templates, 27-30
  Grid App project template, 28
  Hub App project template, 28-30
  Navigation App project template, 27
  Split App project template, 29-30
Windows Store apps, running, 31-33

W
W3C selector standard, 64
warnings, Flyout controls, 169-171
web pages
  embedding in Windows Store app, 139-146
  hosting with WebView control, 140-142
web service data sources, creating, 276-281
Web Workers, 22
WebGL, 22
websites, CommonJS, 57
WebView control, 139-146
  events, 142
  navigation, handling, 142-144
  screenshots, capturing, 145-146
web pages
  hosting, 140-142
winControl property, 99-100
window resize events, 324-325
windows, designing apps for different window sizes, 320
  CSS media queries, 321-324
  setting minimum app width, 320-321
  window resize events, 324-325
Windows 8, migrating to Windows 8.1, 40-42
Windows 8.1, migrating from Windows 8, 40-42
Windows App Certification kit, launching, 40-41
Windows Azure Management Portal, creating Mobile Services, 295-297
Windows Content Indexer API, 381
Windows Developer, registering as, 36
Windows Index, 380
  Indexer Helper, 382-383
  Indexer helper object, creating, 381-382
  querying, 383-384
Windows RT, 23
Windows Store apps, 5-13
  app bar, 8-9
  app capabilities, declaring, 15-17
  charms, 9-11
  closing, 13
  common features of, 7-13
  debugging in Visual Studio, 33-36
    DOM Explorer, 35-36
    JavaScript Console window, 33-34
elements of, 21-26
  CSS3, 22
  HTML5, 22
  JavaScript, 21-22
  Windows RT, 23
  WinJS, 23-24
elements of Windows Store apps
  JQuery, 24-26
Microsoft design style principles, 6-7
  nav bar, 9
orientations, 11-12
running in Visual Studio, 31-33
scopes, specifying, 391-393
screen resolutions, 11-12
sharing across multiple devices, 12-13
Visual Studio project
  creating, 14-15
web pages, embedding, 139-146
Windows Store, publishing to, 36-40
  passing app certification, 39-40
  submitting your app, 37-38
Windows Developer, registering as, 36
win-item class, 199
win-itembox class, 199
win-itemcontainer class, 199
WinJS (Windows Library for JavaScript), 23-24, 45
classes, 51-52
templates, 81
controls, 113-120
  AppBar control, 176-184
  creating declaratively, 115-117
  creating imperatively, 117-118
  DatePicker control, 127-133
  and declarative data binding, 99-100
  declaring, 113-114
  Hub control, 137-139
  ListView control, 223
  Menu control, 172-174
  NavBar control, 184-186
  options, setting, 118-119
  Rating control, 124-127
  references, adding, 114-115
  retrieving from HTML documents, 119-120
  TimePicker control, 133-136
  ToggleSwitch control, 122-124
  ToolTip control, 120-121
  WebView control, 139-146
  modules, 48-51
  namespaces, 46-48
  promises, 56-63
canceling, 62-63
  composing, 63
  creating, 60-61
timeout promises, 61-62
  query selectors, 63-69
  QueryCollection class, 68-69
  WinJS.Utilities.children() method, 67-68
  WinJS.Utilities.id() method, 66-67
  WinJS.Utilities.query() method, 64-66
  Scheduler, 75-80
WinJS library, installing Mobile Services for, 298-299
  WinJS.Application.errors, 312
  WinJS.Application.settings, 312
  WinJS.Application.unload, 311
  WinJS.Binding.List object, 88-90
  WinJS.Class.define() method, 51-52
  WinJS.Class.derive() method, 53-54
  WinJS.Class.mix() method, 54-56
  WinJS.Namespace.define() method, 46-48
  WinJS.Namespace.defineWithParent() method, 48
  WinJS.UI.processAll() method, 20
  WinJS.Utilities.children() method, 67-68
  WinJS.Utilities.id() method, 66-67
  WinJS.Utilities.query() method, 64-66
  WinJS.xhr() method, 58, 69-74

X-Y-Z

xhr() function, performing Ajax calls with, 69-74
  response types, specifying, 72-73
XmlHttpRequest object, 72-74
  years, omitting from DatePicker control, 131-132
yielding to higher priority jobs, 77-80