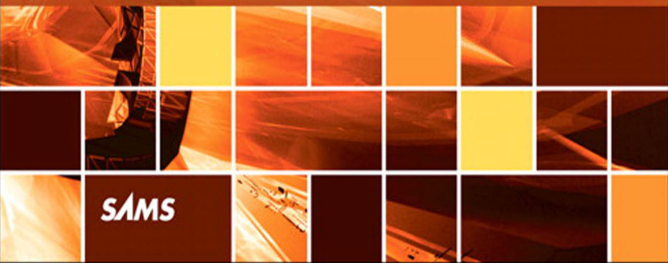
Stephen Walther

Windows® 8.1 Apps

with HTML5 and JavaScript

UNLEASHED



FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Stephen Walther

Windows® 8.1 Apps with HTML5 and JavaScript

UNLEASHED

SAMS | 800 East 96th Street, Indianapolis, Indiana 46240 USA

Windows® 8.1 Apps with HTML5 and JavaScript Unleashed

ISBN-13: 978-0-672-33711-6
ISBN-10: 0-672-33711-8

Library of Congress Control Number 2013951680

Printed in the United States on America

First Printing December 2013

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author(s) and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the programs accompanying it.

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact international@pearsoned.com.

Editor-in-Chief
Greg Wiegand

Executive Editor
Neil Rowe

Development Editor
Mark Renfrow

Managing Editor
Kristy Hart

Project Editor
Elaine Wiley

Indexer
Tim Wright

Proofreader
Charlotte HKughen

Technical Editors
Jeff Burtoft
James Boddie

Publishing Coordinator
Cindy Teeters

Senior Compositor
Gloria Schurick

Contents at a Glance

Table of Contents

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).

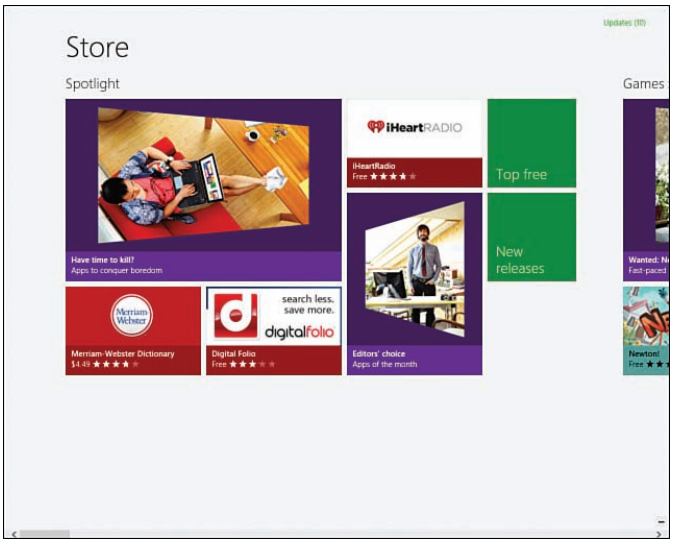


FIGURE I.1 You can sell your app in the Windows Store.

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.

This page intentionally left blank

CHAPTER 1

Building Windows Store Apps

IN THIS CHAPTER

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.

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.

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.

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.

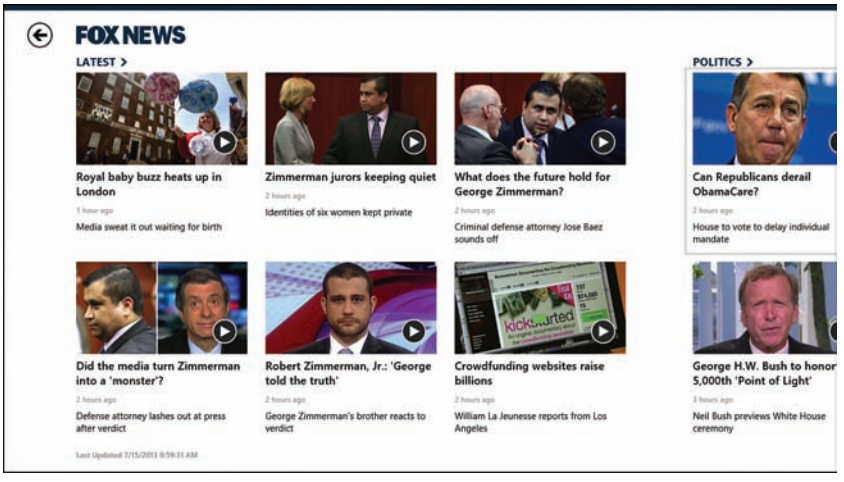


FIGURE 1.1 Windows 8 Bing News app

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.

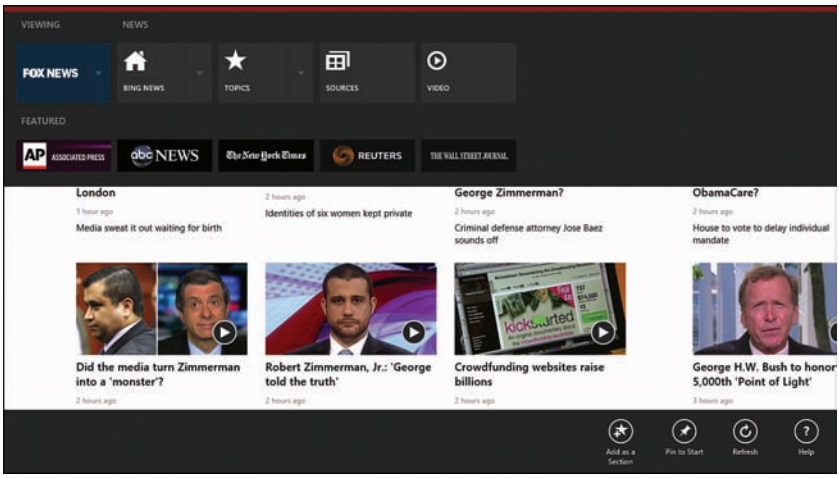


FIGURE 1.2 Using the app bar and nav bar

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).

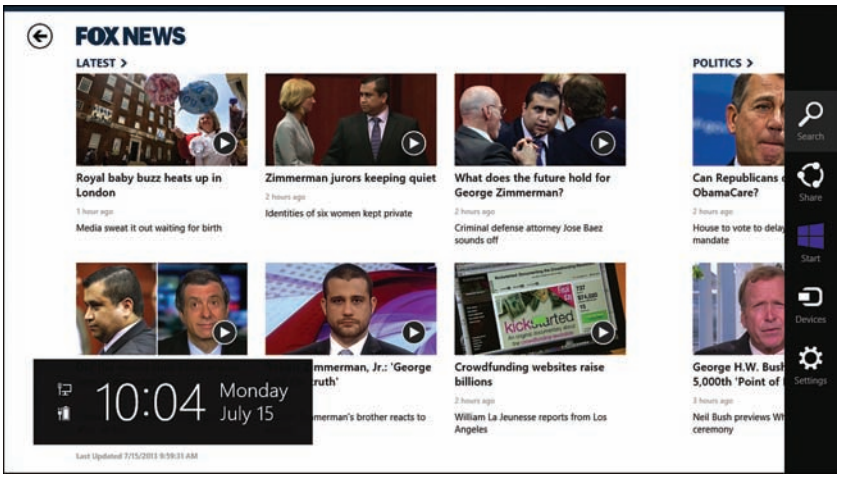


FIGURE 1.3 Viewing charms

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.

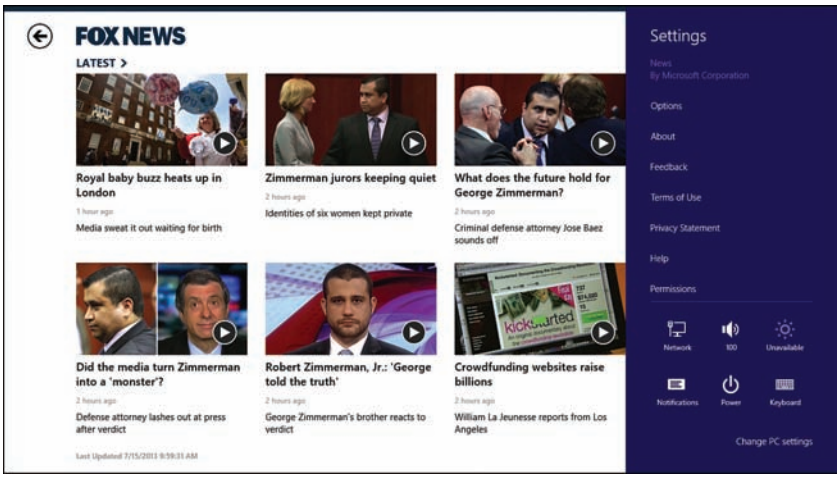


FIGURE 1.4 The Settings charm

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.

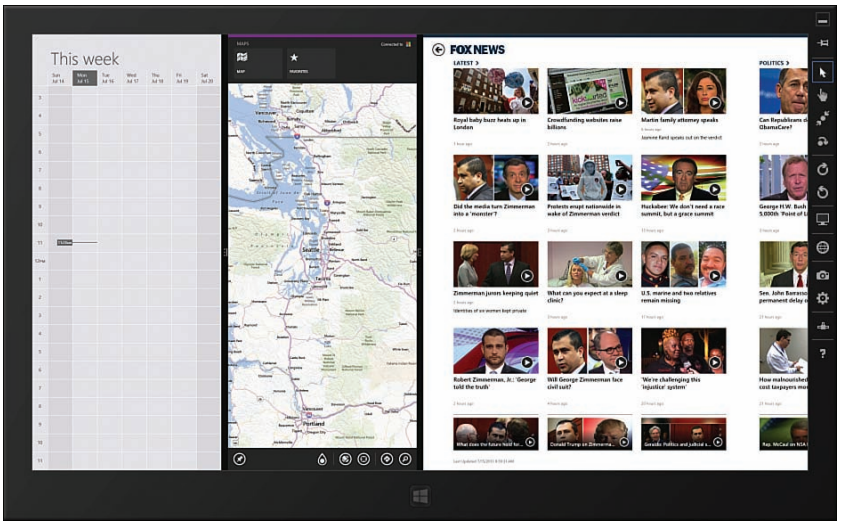


FIGURE 1.5 Three Windows 8.1 apps side by side

NOTE

Windows 8, unlike Windows 8.1, supported running of no more than two apps at once. Furthermore, when using Windows 8, one of the two running apps was required to be snapped to a horizontal resolution of 320 pixels. Windows 8.1 is far more flexible.

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.

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.

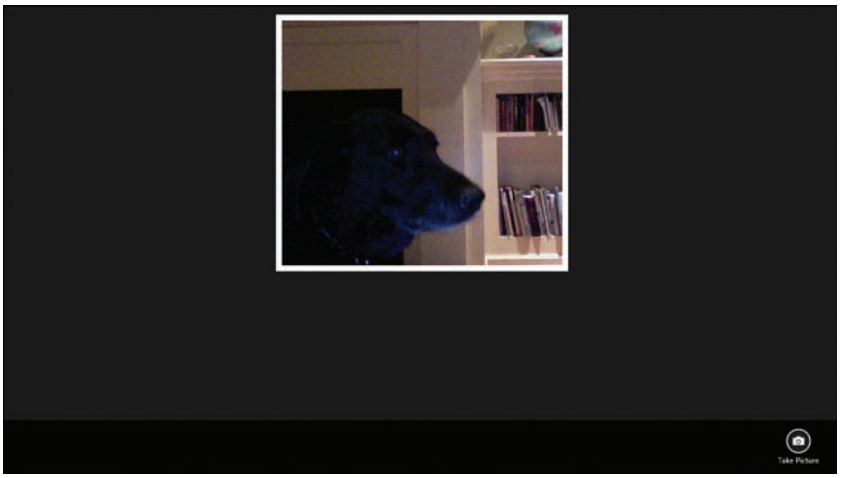


FIGURE 1.6 Your first Windows Store app

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).

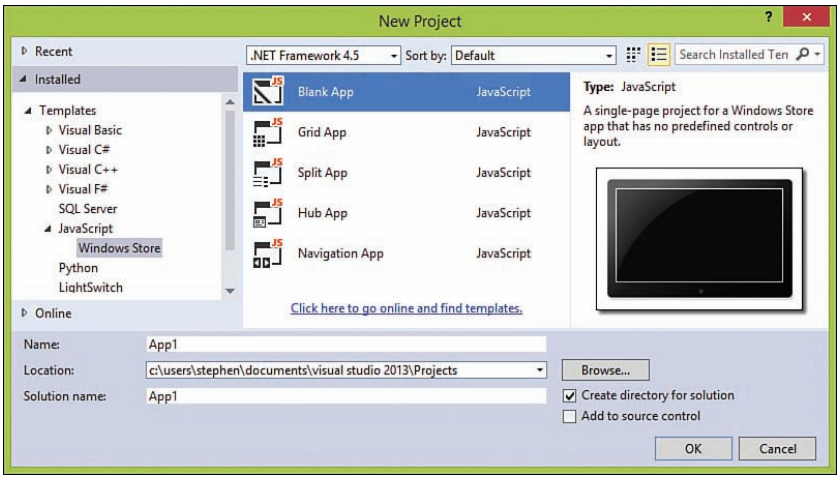


FIGURE 1.7 Using the Visual Studio New Project dialog

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.

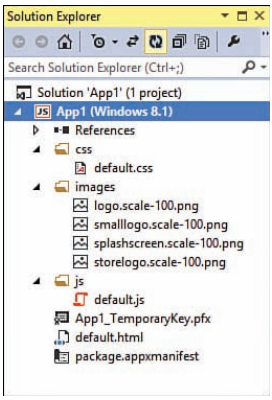


FIGURE 1.8 Windows Store app default files

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).

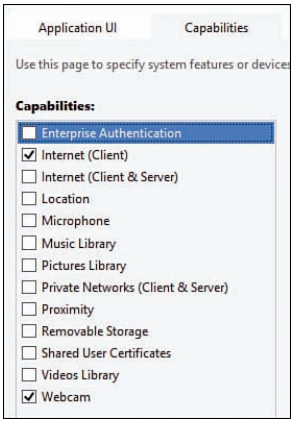


FIGURE 1.9 Enabling the capability to take pictures

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.

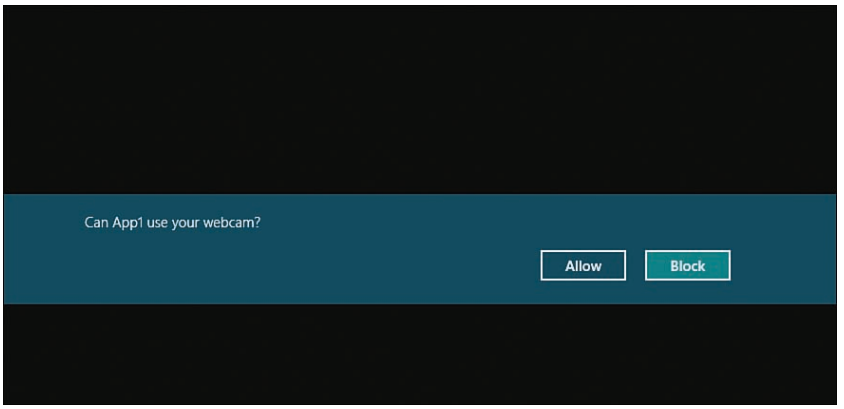


FIGURE 1.10 Asking for consent to access your webcam

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

```
<!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).

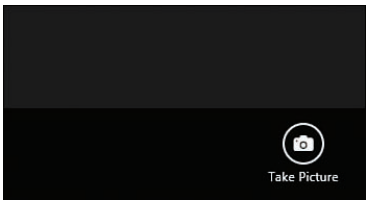


FIGURE 1.11 The Take Picture command in the app bar

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:

```
#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

```
(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.

FIGURE 1.12 The camera capture UI screen

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.

The photo is displayed in the HTML page with the following line of code:

```
// 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.

FIGURE 1.13 Running a Windows Store app

Assuming that your laptop or tablet has a camera, you can start taking pictures.

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:

```
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:

```
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:

- **Geolocation**—Use the WinRT `Windows.Devices.Geolocation.Geolocator` class to get your current latitude and longitude.
- **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.

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).

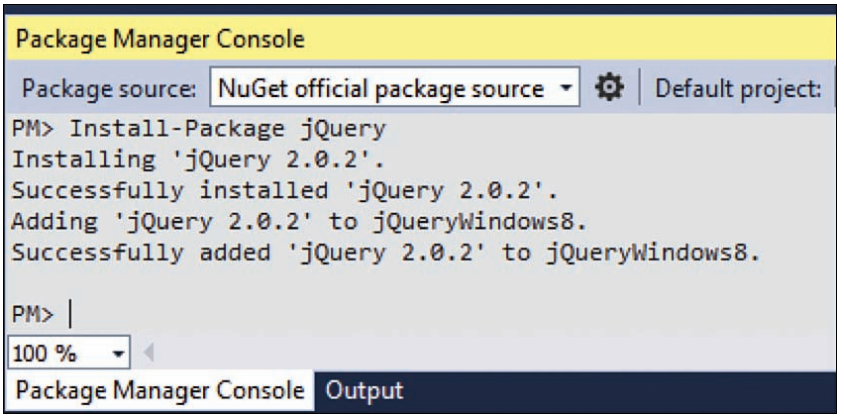


FIGURE 1.14 Adding jQuery with the Library Package Manager Console

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.

LISTING 1.3 Using jQuery in a Windows Store App

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>jQueryWindows8</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>

    <!-- jQueryWindows8 references -->
    <script type="text/javascript" src="Scripts/jquery-2.0.2.js"></script>

    <style type="text/css">
        #divMessage {
            display:none;
            padding:10px;
            border: solid 1px white;
            background-color: #ff6a00;
        }
    </style>
</head>
<body>
    <button id="btnShow">Click Here</button>
    <div id="divMessage">
        Secret Message
    </div>

    <script type="text/javascript">
        $("#btnShow").click(function () {
            $("#divMessage").fadeToggle("slow");
        });
    </script>

</body>
</html>
```

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).

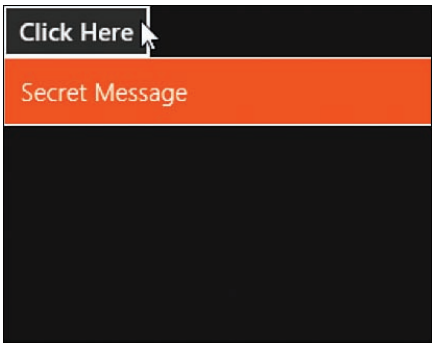


FIGURE 1.15 Using jQuery to animate a DIV element

NOTE

The code in Listing 1.3 is contained in the Chapter 1 folder in a folder named jQueryWindows8.

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:

1. Blank App—The simplest of the templates. Contains a single default.html, default.css, and default.js file.
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."

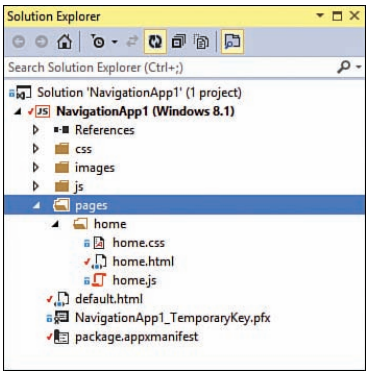


FIGURE 1.16 Creating a multi-page app with the Navigation App project template

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).

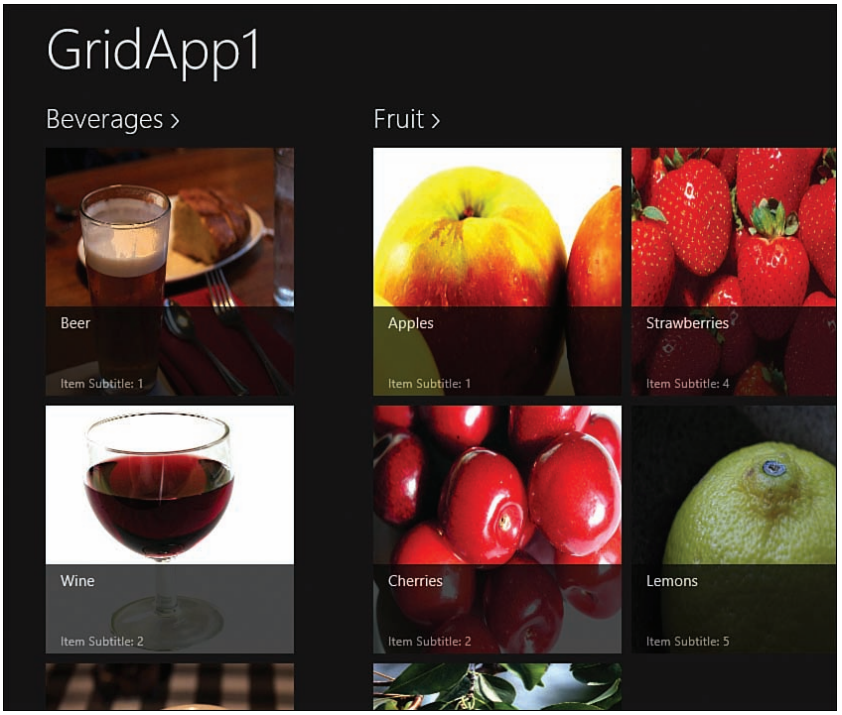


FIGURE 1.17 The Grid App groupedItems page

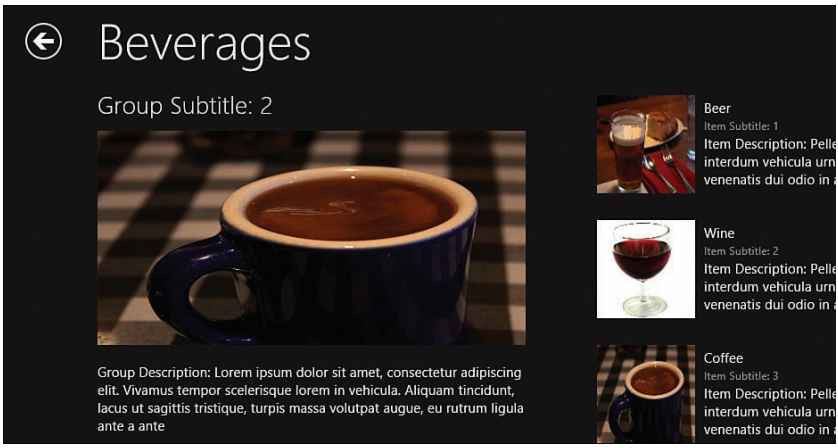


FIGURE 1.18 The Grid App groupDetail page

FIGURE 1.19 The Grid App itemDetail 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.

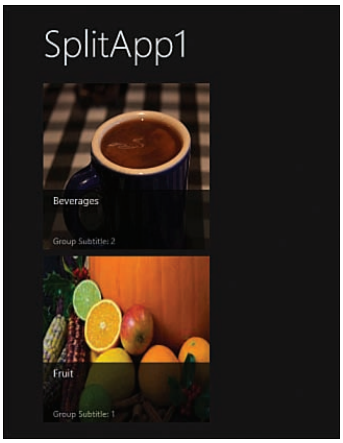


FIGURE 1.20 The Split App items page

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).

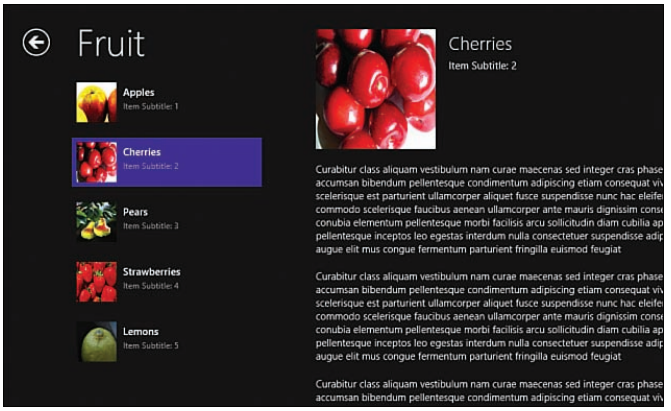


FIGURE 1.21 The Split App split page

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.

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).

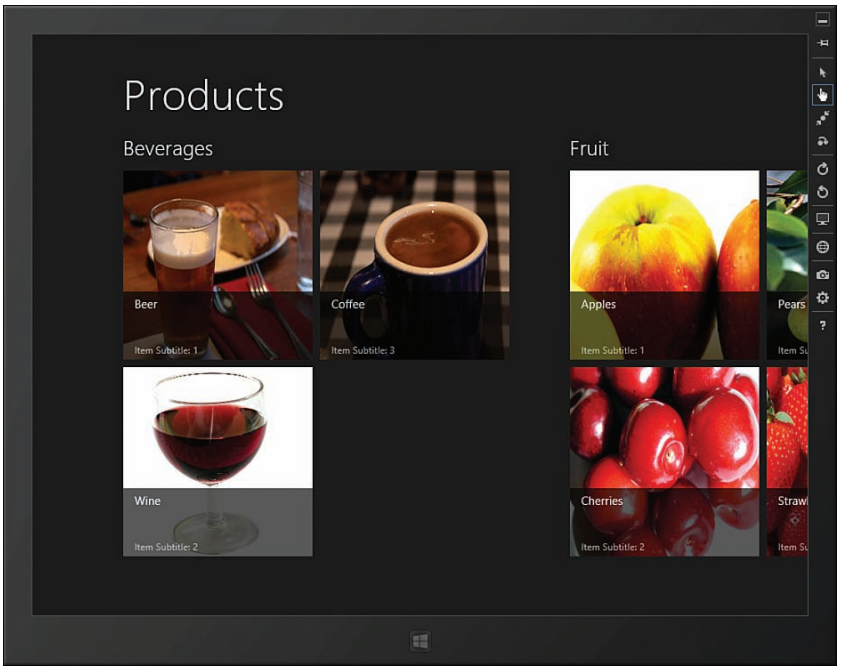


FIGURE 1.23 Using the Visual Studio simulator

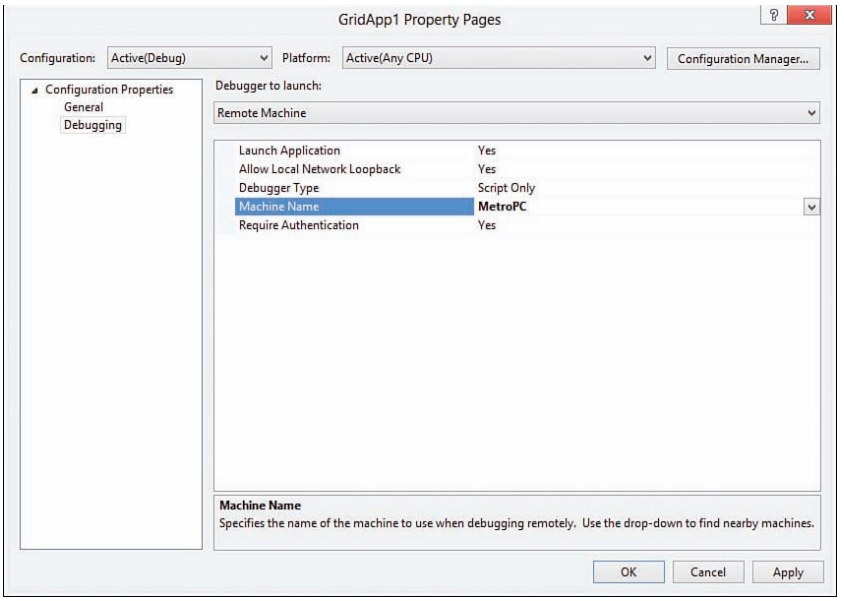


FIGURE 1.24 Specifying the remote machine name

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.)

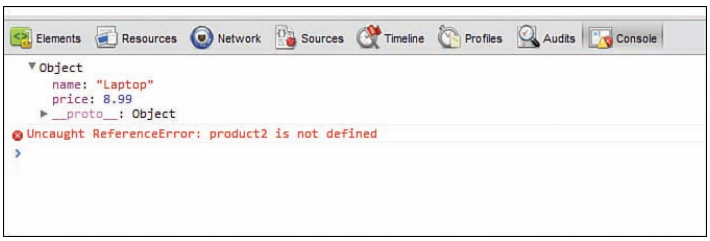


FIGURE 1.25 Debugging with the Google Chrome JavaScript console

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).

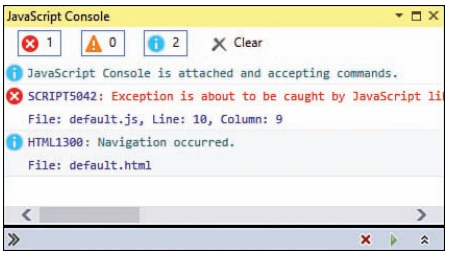


FIGURE 1.26 The Visual Studio JavaScript Console Window

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).

FIGURE 1.27 Dumping a JavaScript variable to the JavaScript Console window

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.

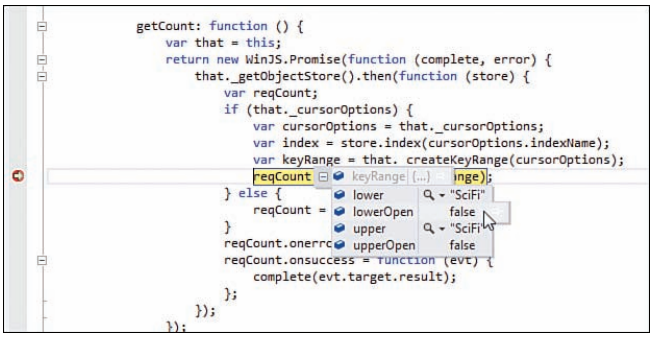


FIGURE 1.28 Setting a breakpoint

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).

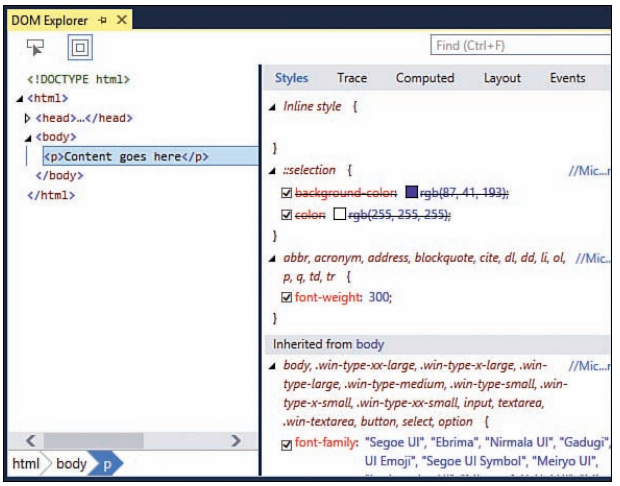


FIGURE 1.29 Using the DOM Explorer Window

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

You can distribute your app without publishing to the Windows Store by taking advantage of a feature called *sideloading*. In order to take advantage of sideloading, you must sign your app and configure the right group policy settings on the target computers. You can learn about sideloading by visiting http://technet.microsoft.com/en-us/library/hh852635.aspx.

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.

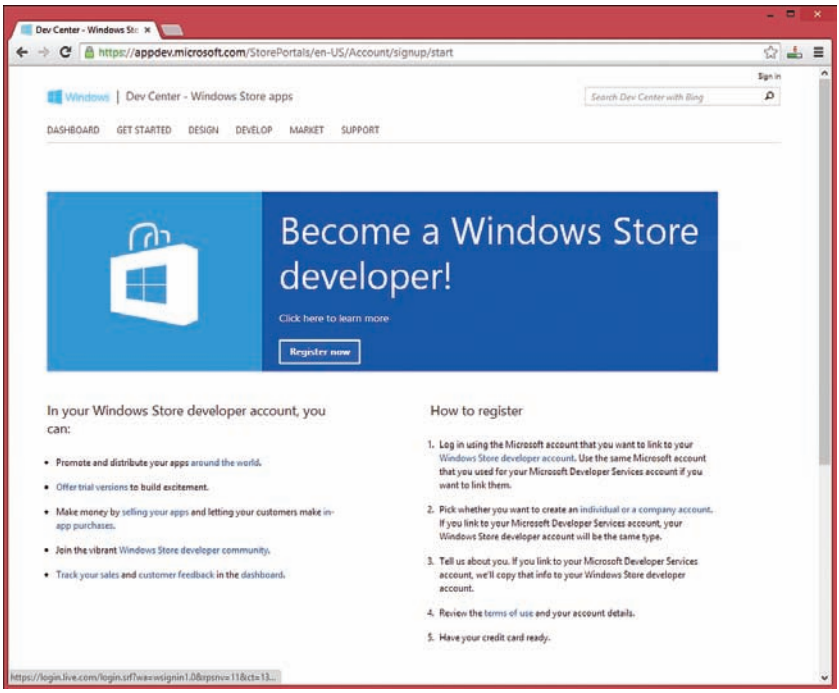


FIGURE 1.30 Register as a Windows Store developer

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.

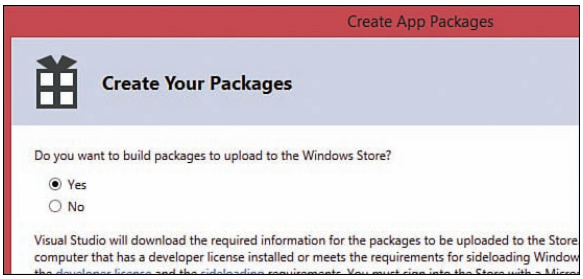


FIGURE 1.32 Creating your app package

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

The Windows Store certification requirements are detailed at http://msdn.microsoft.com/en-us/library/windows/apps/hh694083.aspx.

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).

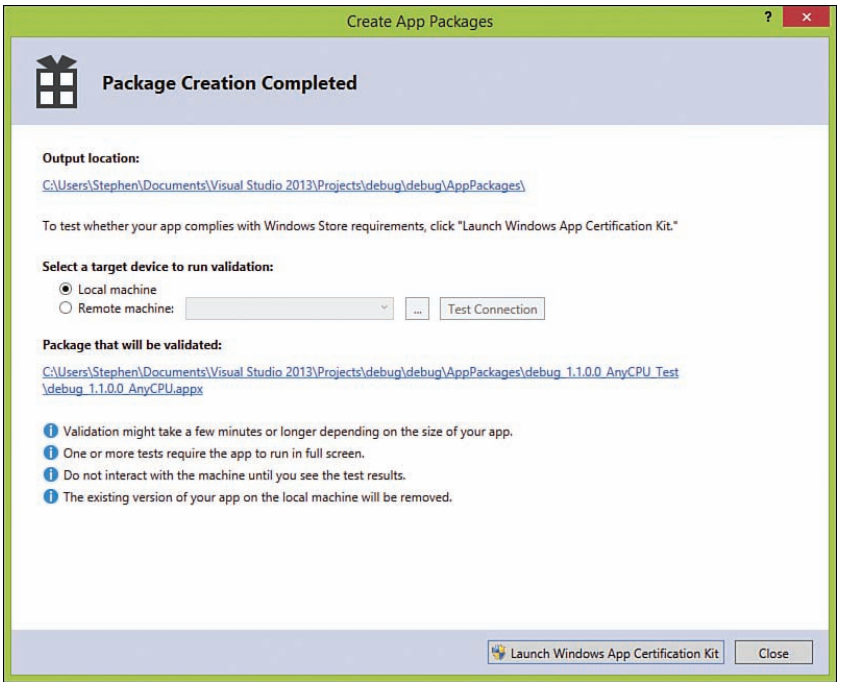


FIGURE 1.33 Launching the Windows App Certification Kit

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.

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).

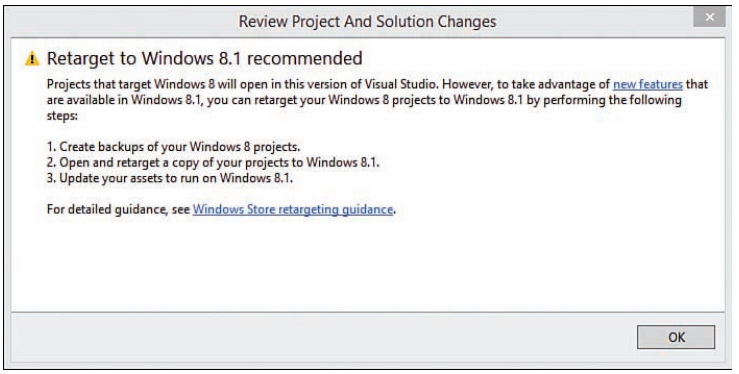


FIGURE 1.35 Retargeting to Windows 8.1

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

Index

Symbols

A

B

C

D

E

F

G

H

I

J-K

L

M

N

O

P

Q

R

S

T

U

V

W

X-Y-Z