Windows® 8 Apps with XAML and C#
Adam Nathan

Windows® 8 Apps with XAML and C# Unleashed
# Contents at a Glance

## Introduction

1. Introduction

## Part I  Getting Started

1. Anatomy of a Windows Store App
   - Page 7
2. Mastering XAML
   - Page 27

## Part II  Building an App

3. Sizing, Positioning, and Transforming Elements
   - Page 47
4. Layout
   - Page 65
5. Interactivity
   - Page 101
6. Handling Input: Touch, Mouse, Pen, and Keyboard
   - Page 115
7. App Model
   - Page 149

## Part III  Understanding Controls

8. Content Controls
   - Page 185
9. Items Controls
   - Page 207
10. Text
    - Page 227
11. Images
    - Page 253
12. Audio and Video
    - Page 285
13. Other Controls
    - Page 313

## Part IV  Leveraging the Richness of XAML

14. Vector Graphics
    - Page 333
15. Animation
    - Page 365
16. Styles, Templates, and Visual States
    - Page 409
Table of Contents

Introduction 1
Who Should Read This Book? 3
Software Requirements 3
Code Examples 3
How This Book Is Organized 3
Conventions Used in This Book 5

Part I  Getting Started

1 Anatomy of a Windows Store App 7
   Launching a New App 8
   The Package Manifest 9
   The Main Page 19
   The Application Definition 21
   Summary 25

2 Mastering XAML 27
   Elements and Attributes 28
   Namespaces 29
   Property Elements 31
   Type Converters 33
   Markup Extensions 34
   Children of Object Elements 36
   Mixing XAML with Procedural Code 40
   XAML Keywords 44
   Summary 45

Part II  Building an App

3 Sizing, Positioning, and Transforming Elements 47
   Controlling Size 48
   Controlling Position 52

   Applying 2D Transforms 55
   Applying 3D Transforms 62
   Summary 64

4 Layout 65
   Discovering the Current Dimensions 66
   Discovering the Current View State 67
   Discovering the Current Orientation 70
   Panels 71
   Handling Content Overflow 87
   Summary 99

5 Interactivity 101
   Dependency Properties 101
   Routed Events 108
   Commands 113
   Summary 114

6 Handling Input: Touch, Mouse, Pen, and Keyboard 115
   Touch Input 116
   Mouse Input 138
   Pen Input 140
   Keyboard Input 142
   Summary 147

7 App Model 149
   Understanding the App Lifecycle 150
   Programmatically Launching Apps 163
## Table of Contents

### Part III Understanding Controls

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 Content Controls</td>
<td>185</td>
</tr>
<tr>
<td>Button</td>
<td>188</td>
</tr>
<tr>
<td>HyperlinkButton</td>
<td>189</td>
</tr>
<tr>
<td>RepeatButton</td>
<td>191</td>
</tr>
<tr>
<td>ToggleButton</td>
<td>191</td>
</tr>
<tr>
<td>CheckBox</td>
<td>192</td>
</tr>
<tr>
<td>RadioButton</td>
<td>192</td>
</tr>
<tr>
<td>ToolTip</td>
<td>194</td>
</tr>
<tr>
<td>AppBar</td>
<td>196</td>
</tr>
<tr>
<td>Summary</td>
<td>205</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 Items Controls</td>
<td>207</td>
</tr>
<tr>
<td>Items in the Control</td>
<td>208</td>
</tr>
<tr>
<td>Items Panels</td>
<td>210</td>
</tr>
<tr>
<td>ComboBox</td>
<td>213</td>
</tr>
<tr>
<td>ListBox</td>
<td>214</td>
</tr>
<tr>
<td>ListView</td>
<td>216</td>
</tr>
<tr>
<td>GridView</td>
<td>219</td>
</tr>
<tr>
<td>FlipView</td>
<td>221</td>
</tr>
<tr>
<td>SemanticZoom</td>
<td>223</td>
</tr>
<tr>
<td>Summary</td>
<td>226</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 Text</td>
<td>227</td>
</tr>
<tr>
<td>TextBlock</td>
<td>227</td>
</tr>
<tr>
<td>RichTextBlock</td>
<td>235</td>
</tr>
<tr>
<td>TextBox</td>
<td>240</td>
</tr>
<tr>
<td>RichEditBox</td>
<td>248</td>
</tr>
<tr>
<td>PasswordBox</td>
<td>251</td>
</tr>
<tr>
<td>Summary</td>
<td>252</td>
</tr>
</tbody>
</table>

### Part IV Leveraging the Richness of XAML

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 Images</td>
<td>253</td>
</tr>
<tr>
<td>The Image Element</td>
<td>253</td>
</tr>
<tr>
<td>Multiple Files for Multiple Environments</td>
<td>263</td>
</tr>
<tr>
<td>Decoding Images</td>
<td>267</td>
</tr>
<tr>
<td>Encoding Images</td>
<td>276</td>
</tr>
<tr>
<td>Summary</td>
<td>284</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 Audio and Video</td>
<td>285</td>
</tr>
<tr>
<td>Playback</td>
<td>286</td>
</tr>
<tr>
<td>Capture</td>
<td>294</td>
</tr>
<tr>
<td>Transcoding</td>
<td>305</td>
</tr>
<tr>
<td>Summary</td>
<td>311</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13 Other Controls</td>
<td>313</td>
</tr>
<tr>
<td>Range Controls</td>
<td>313</td>
</tr>
<tr>
<td>Popup Controls</td>
<td>316</td>
</tr>
<tr>
<td>A Few More Controls</td>
<td>325</td>
</tr>
<tr>
<td>Summary</td>
<td>330</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>14 Vector Graphics</td>
<td>333</td>
</tr>
<tr>
<td>Shapes</td>
<td>334</td>
</tr>
<tr>
<td>Geometries</td>
<td>340</td>
</tr>
<tr>
<td>Brushes</td>
<td>348</td>
</tr>
<tr>
<td>Summary</td>
<td>363</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 Animation</td>
<td>365</td>
</tr>
<tr>
<td>Theme Transitions</td>
<td>366</td>
</tr>
<tr>
<td>Theme Animations</td>
<td>376</td>
</tr>
<tr>
<td>Custom Animations</td>
<td>382</td>
</tr>
<tr>
<td>Custom Keyframe Animations</td>
<td>395</td>
</tr>
<tr>
<td>Easing Functions</td>
<td>400</td>
</tr>
<tr>
<td>Manual Animations</td>
<td>404</td>
</tr>
<tr>
<td>Summary</td>
<td>406</td>
</tr>
</tbody>
</table>
## Part V Exploiting Windows 8

### 17 Data Binding 439
- Introducing Binding 439
- Controlling Rendering 447
- Customizing the View of a Collection 455
- Summary 459

### 18 Data 461
- App Data 461
- User Data 466
- Networking 469
- Summary 474

### 19 Charms 477
- Search 477
- Share 486
- Devices 492
- Settings 503
- Summary 508

### 20 Extensions 509
- Account Picture Provider 509
- AutoPlay Content and AutoPlay Device 512
- Contact Picker 514
- File Type Associations 516
- Protocol 518
- Background Tasks 519
- Summary 527

### 21 Sensors and Other Devices 529
- Accelerometer 529
- Gyrometer 532
- Inclinometer 532
- Compass 533
- Light Sensor 533
- Orientation 533
- Location 534
- Proximity 535
- Summary 538

## Part VI Advanced Topics

### 22 Thinking Outside the App: Live Tiles, Toast Notifications, and the Lock Screen 539
- Live Tiles 539
- Toast Notifications 552
- The Lock Screen 556
- Summary 557

Index 559
About the Author

Adam Nathan is a principal software architect for Microsoft, a best-selling technical author, and arguably the world’s most prolific developer for Windows Phone. He introduced XAML to countless developers through his books on a variety of Microsoft technologies. Currently a part of Microsoft’s Startup Business Group, Adam has previously worked on Visual Studio and the Common Language Runtime. He was the founding developer and architect of Popfly, Microsoft’s first Silverlight-based product, named by PCWorld as one of its year’s most innovative products. He is also the founder of PINVOKE.NET, the online resource for .NET developers who need to access Win32. His apps have been featured on Lifehacker, Gizmodo, ZDNet, ParentMap, and other enthusiast sites.

Adam’s books are considered required reading by many inside Microsoft and throughout the industry. Adam is the author of 101 Windows Phone 7 Apps (Sams, 2011), Silverlight 1.0 Unleashed (Sams, 2008), WPF Unleashed (Sams, 2006), WPF 4 Unleashed (Sams, 2010), and .NET and COM: The Complete Interoperability Guide (Sams, 2002); a coauthor of ASP.NET: Tips, Tutorials, and Code (Sams, 2001); and a contributor to books including .NET Framework Standard Library Annotated Reference, Volume 2 (Addison-Wesley, 2005) and Windows Developer Power Tools (O’Reilly, 2006). You can find Adam online at www.adamnathan.net, or @adamnathan on Twitter.
Dedication

To Tyler and Ryan.
Acknowledgments

First, I thank Lindsay Nathan for making this possible. Words fail to describe my gratitude.

I’d like to give special thanks to Ashish Shetty, Tim Heuer, Mark Rideout, Jonathan Russ, Joe Duffy, Chris Brumme, Eric Rudder, Neil Rowe, Betsy Harris, Ginny Munroe, Eileen Chan, and Valery Sarkisov. As always, I thank my parents for having the foresight to introduce me to Basic programming on our IBM PCjr when I was in elementary school.

Finally, I thank you for picking up a copy of this book! I don’t think you’ll regret it!
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.

E-mail: 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 ask me, it has never been a better time to be a software developer. Not only are programmers in high demand—due in part to an astonishingly low number of computer science graduates each year—but app stores make it easier than ever to broadly distribute your own software and even make money from it!

I remember releasing a few shareware games in junior high school and asking for $5 donations. I earned $15. One of the three donations was from my grandmother, who didn’t even own a computer! These days, of course, adults and kids alike can make money on simple apps and games without relying on kind and generous individuals going to the trouble of mailing a check!

The Windows Store is an app store like no other. When you consider the number of people who use Windows 8 (and Windows RT) compared to the number of people who use any other operating system on the planet, you realize what a unique and enormous opportunity the Windows Store provides.

When you write a Windows Store app, you can work with whichever language and technology is most comfortable for you: JavaScript with an HTML user interface, or C#/Visual Basic/C++ with a XAML or raw DirectX user interface. (You can also componentize code to get different mixtures, such as using C# with HTML or some JavaScript in a XAML app.) Besides familiarity, your choice can have other benefits. Outside of the core Windows platform, each language and technology has different sets of reusable libraries and components. C++ has features for high-performance algorithms, for example. However, regardless of which choice you make, the Windows APIs are the same, and the graphics are hardware accelerated.
The key to the multiple-language support is the Windows Runtime, or WinRT for short. You can think of it like .NET’s Common Language Runtime, except it spans both managed and unmanaged languages. To enable this, WinRT is COM-based. Most of the time, you can’t tell when you interact with WinRT, however. This is a new, friendlier version of COM that is more amenable to automatic correct usage from environments such as .NET or JavaScript. (Contrast this to over a decade ago, when I wrote a book about mixing COM with .NET. This topic alone required over 1,600 pages!)

WinRT APIs are automatically projected into the programming language you use, so they look natural for that language. Projections are more than just exposing the raw APIs, however. Core WinRT data types such as string, collection types, and a few others are mapped to appropriate data types for the target environment. For C# or other .NET languages, this means exposing them as System.String, System.Collections.Generic.IList<T>, and so on. To match conventions, member names are even morphed to be Camel-cased for JavaScript and Pascal-cased for other languages, which makes the MSDN reference documentation occasionally look goofy.

In the set of APIs exposed by Windows, everything under the Windows.UI.Xaml namespace is XAML-specific, everything under the Windows.UI.WebUI namespace is for HTML apps, everything under System is .NET-specific, and everything else (which is under Windows) is general-purpose WinRT functionality. As you dig into the framework, you notice that the XAML-specific and .NET-specific APIs are indeed the most natural to use from C# and XAML. General-purpose WinRT APIs follow slightly different conventions and can sometimes look a little odd to developers familiar with .NET. For example, they tend to be exception-heavy for situations that normally don’t warrant an exception (such as the user cancelling an action). Artifacts like this are caused by the projection mechanism mapping HRESULTs (COM error codes) into .NET exceptions.

I wrote this book with the following goals in mind:

➔ To provide a solid grounding in the underlying concepts, in a practical and approachable fashion
➔ To answer the questions most people have when learning how to write Windows Store apps and to show how commonly desired tasks are accomplished
➔ To be an authoritative source, thanks to input from members of the team who designed, implemented, and tested Windows 8 and Visual Studio
➔ To be clear about where the technology falls short rather than blindly singing its praises
To optimize for concise, easy-to-understand code rather than enforcing architectural patterns that can be impractical or increase the number of concepts to understand.

To be an easily navigated reference that you can constantly come back to.

To elaborate on the second-to-last point: You won’t find examples of patterns such as Model-View-ViewModel (MVVM) in this book. I am a fan of applying such patterns to code, but I don’t want to distract from the core lessons in each chapter.

Whether you’re new to XAML or a long-time XAML developer, I hope you find this book to exhibit all these attributes.

Who Should Read This Book?

This book is for software developers who are interested in creating apps for the Windows Store, whether they are for tablets, laptops, or desktops. It does not teach you how to program, nor does it teach the basics of the C# language. However, it is designed to be understandable even for folks who are new to .NET, and does not require previous experience with XAML. And if you are already well versed in XAML, I’m confident that this book still has a lot of helpful information for you. At the very least, it should be an invaluable reference for your bookshelf.

Software Requirements

This book targets Windows 8, Windows RT, and the corresponding developer tools. The tools can be downloaded for free at the Windows Dev Center: s. The download includes the Windows 8 SDK, a version of Visual Studio Express specifically for Windows Store apps, and Blend. It’s worth noting that although this book almost exclusively refers to Windows 8, the content also applies to Windows RT.

Although it’s not required, I recommend PAINT.NET, a free download at http://getpaint.net, for creating and editing graphics, such as the set of icons needed by apps.

Code Examples


How This Book Is Organized

This book is arranged into five parts, representing the progression of feature areas that you typically need to understand. But if you want to jump ahead and learn about a topic such as animation or live tiles, the book is set up to allow for nonlinear journeys as well. The following sections provide a summary of each part.
Part I: Getting Started
This part includes the following chapters:
➔ Chapter 1, “Anatomy of a Windows Store App”
➔ Chapter 2, “Mastering XAML”

This part provides the foundation for the rest of the book. If you have previously created Windows Phone apps or worked with XAML in the context of other Microsoft technologies, a lot of this should be familiar to you. There are still several unique aspects for Windows 8 and the Windows Store, however.

Part II: Building an App
This part includes the following chapters:
➔ Chapter 3, “Sizing, Positioning, and Transforming Elements”
➔ Chapter 4, “Layout”
➔ Chapter 5, “Interactivity”
➔ Chapter 6, “Handling Input: Touch, Mouse, Pen, and Keyboard”
➔ Chapter 7, “App Model”

Part II equips you with the knowledge of how to place things on the screen, how to make them adjust to the wide variety of screen types, and how to interact with the user. It also digs into the app model for Windows Store apps, which is significantly different from the app model for desktop applications in a number of ways.

Part III: Understanding Controls
This part includes the following chapters:
➔ Chapter 8, “Content Controls”
➔ Chapter 9, “Items Controls”
➔ Chapter 10, “Text”
➔ Chapter 11, “Images”
➔ Chapter 12, “Audio and Video”
➔ Chapter 13, “Other Controls”

Part III provides a tour of the controls built into the XAML UI Framework. There are many controls that you expect to have available, plus several that you might not expect.
Part IV: Leveraging the Richness of XAML
This part includes the following chapters:

➔ Chapter 14, “Vector Graphics”
➔ Chapter 15, “Animation”
➔ Chapter 16, “Styles, Templates, and Visual States”
➔ Chapter 17, “Data Binding”

The features covered in Part IV are areas in which XAML really shines. Although previous parts of the book expose some XAML richness (applying transforms to any elements, the compositability of controls, and so on), these features push the richness to the next level.

Part V: Exploiting Windows 8
This part includes the following chapters:

➔ Chapter 18, “Data”
➔ Chapter 19, “Charms”
➔ Chapter 20, “Extensions”
➔ Chapter 21, “Sensors and Other Devices”
➔ Chapter 22, “Thinking Outside the App: Live Tiles, Toast Notifications, and the Lock Screen”

This part of the book can just as easily appear in a book about JavaScript or C++ Windows Store apps, with the exception of its code snippets. It covers unique and powerful Windows 8 features that are not specific to XAML or C#, but they are things that all Windows Store app developers should know.

Conventions Used in This Book
Various typefaces in this book identify new terms and other special items. These typefaces include the following:

<table>
<thead>
<tr>
<th>Typeface</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Italic</em></td>
<td>Italic is used for new terms or phrases when they are initially defined and occasionally for emphasis.</td>
</tr>
<tr>
<td>Monospace</td>
<td>Monospace is used for screen messages, code listings, and filenames. In code listings, <em>italic monospace type</em> is used for placeholder text. Code listings are colorized similarly to the way they are colorized in Visual Studio. <em>Blue monospace type</em> is used for XML elements and C# keywords, <em>brown monospace type</em> is used for XML element names and C# strings, <em>green monospace type</em> is used for comments, <em>red monospace type</em> is used for XML attributes, and <em>teal monospace type</em> is used for type names in C#.</td>
</tr>
<tr>
<td><strong>Bold</strong></td>
<td>When appropriate, bold is used for code directly related to the main lesson(s) in a chapter.</td>
</tr>
</tbody>
</table>
Throughout this book, and even in this introduction, you find a number of sidebar elements:

**What is a FAQ sidebar?**
A Frequently Asked Question (FAQ) sidebar presents a question you might have about the subject matter in a particular spot in the book—and then provides a concise answer.

**Digging Deeper Sidebars**
A Digging Deeper sidebar presents advanced or more detailed information on a subject than is provided in the surrounding text. Think of Digging Deeper material as something you can look into if you’re curious but can ignore if you’re not.

**A tip offers information about design guidelines, shortcuts, or alternative approaches to produce better results, or something that makes a task easier.**

**Warning!**
A warning alerts you to an action or a condition that can lead to an unexpected or unpredictable result—and then tells you how to avoid it.
You might be thinking, “Isn’t Chapter 2 a bit early to become a master of XAML?” No, because this chapter focuses on the mechanics of the XAML language, which is a bit orthogonal to the multitude of XAML elements and APIs you’ll be using when you build Windows Store apps. Learning about the XAML language is kind of like learning the features of C# before delving into .NET or the Windows Runtime. Unlike the preceding chapter, this is a fairly deep dive! However, having this background knowledge before proceeding with the rest of the book will enable you to approach the examples with confidence.

XAML is a dialect of XML that Microsoft introduced in 2006 along with the first version of Windows Presentation Foundation (WPF). XAML is a relatively simple and general-purpose declarative programming language suitable for constructing and initializing objects. XAML is just XML, but with a set of rules about its elements and attributes and their mapping to objects, their properties, and the values of those properties (among other things).

You can think of XAML as a clean, modern (albeit more verbose) reinvention of HTML and CSS. In Windows Store apps, XAML serves essentially the same purpose as HTML: It provides a declarative way to represent user interfaces. That said, XAML is actually a general-purpose language that can be used in ways that have nothing to do with UI. The preceding chapter contained a simple example of this. App.xaml does not define a user interface, but rather some characteristics of an app’s entry point class. Note that
almost everything that can be expressed in XAML can be naturally represented in a procedural language like C# as well.

The motivation for XAML is pretty much the same as any declarative markup language: Make it easy for programmers to work with others (perhaps graphic designers) and enable a powerful, robust tooling experience on top of it. XAML encourages a nice separation between visuals (and visual behavior such as animations) and the rest of the code, and enables powerful styling capabilities. XAML pages can be opened in Blend as well as Visual Studio (and Visual Studio has a convenient “Open in Blend...” item on its View menu), or entire XAML-based projects can be opened in Blend. This can be helpful for designing sophisticated artwork, animations, and other graphically rich touches. The idea is that a team’s developers can work in Visual Studio while its designers work in Blend, and everyone can work on the same codebase. However, because XAML (and XML in general) is generally human readable, you can accomplish quite a bit with nothing more than a tool such as Notepad.

**Elements and Attributes**

The XAML specification defines rules that map object-oriented namespaces, types, properties, and events into XML namespaces, elements, and attributes. You can see this by examining the following simple XAML snippet that declares a `Button` control and comparing it to the equivalent C# code:

**XAML:**
```
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    Content="Stop"/>
```

**C#:**
```
b.Content = "Stop";
```

Declaring an XML element in XAML (known as an object **element**) is equivalent to instantiating the corresponding object via a default constructor. Setting an attribute on the object element is equivalent to setting a property of the same name (called a **property attribute**) or hooking up an event handler of the same name (called an **event attribute**). For example, here’s an update to the `Button` control that not only sets its `Content` property but also attaches an event handler to its `Click` event:

**XAML:**
```
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    Content="Stop" Click="Button_Click"/>
```
C#:

```csharp
b.Click += new Windows.UI.Xaml.RoutedEventHandler(Button_Click);
b.Content = "Stop";
```

This requires an appropriate method called `Button_Click` to be defined. The “Mixing XAML with Procedural Code” section at the end of this chapter explains how to work with XAML that requires additional code. Note that XAML, like C#, is a case-sensitive language.

### Order of Property and Event Processing

At runtime, event handlers are always attached before any properties are set for any object declared in XAML (excluding the `Name` property, described later in this chapter, which is set immediately after object construction). This enables appropriate events to be raised in response to properties being set without worrying about the order of attributes used in XAML.

The ordering of multiple property sets and multiple event handler attachments is usually performed in the relative order that property attributes and event attributes are specified on the object element. Fortunately, this ordering shouldn’t matter in practice because design guidelines dictate that classes should allow properties to be set in any order, and the same holds true for attaching event handlers.

### Namespaces

The most mysterious part about comparing the previous XAML examples with the equivalent C# examples is how the XML namespace `http://schemas.microsoft.com/winfx/2006/xaml/presentation` maps to the Windows Runtime namespace `Windows.UI.Xaml.Controls`. It turns out that the mapping to this and other namespaces is hard-coded inside the framework. (In case you’re wondering, no web page exists at the `schemas.microsoft.com` URL—it’s just an arbitrary string like any namespace.) Because many Windows Runtime namespaces are mapped to the same XML namespace, the framework designers took care not to introduce two classes with the same name, despite the fact that the classes are in separate Windows Runtime namespaces.

The root object element in a XAML file must specify at least one XML namespace that is used to qualify itself and any child elements. You can declare additional XML namespaces (on the root or on children), but each one must be given a distinct prefix to be used on any identifiers from that namespace. `MainPage.xaml` in the preceding chapter contains the XML namespaces listed in Table 2.1.
### Table 2.1 The XML Namespaces in Chapter 1’s MainPage.xaml

<table>
<thead>
<tr>
<th>Namespace</th>
<th>Typical Prefix</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://schemas.microsoft.com/winfx/2006/xaml/presentation">http://schemas.microsoft.com/winfx/2006/xaml/presentation</a></td>
<td>(none)</td>
<td>The standard UI namespace. Contains elements such as Grid, Button, and TextBlock.</td>
</tr>
<tr>
<td><a href="http://schemas.microsoft.com/winfx/2006/xaml">http://schemas.microsoft.com/winfx/2006/xaml</a></td>
<td>x</td>
<td>The XAML language namespace. Contains keywords such as Class, Name, and Key.</td>
</tr>
<tr>
<td>using:BlankApp</td>
<td>local</td>
<td>This using:XXX syntax is the way to use any custom Windows Runtime or .NET namespace in a XAML file. In this case, BlankApp is the .NET namespace generated for the project in Chapter 1 because the project itself was named “BlankApp.”</td>
</tr>
<tr>
<td><a href="http://schemas.microsoft.com/expression/blend/2008">http://schemas.microsoft.com/expression/blend/2008</a></td>
<td>d</td>
<td>A namespace for design-time information that helps tools like Blend and Visual Studio show a proper preview.</td>
</tr>
<tr>
<td><a href="http://schemas.openxmlformats.org/microsoft/2006">http://schemas.openxmlformats.org/microsoft/2006</a></td>
<td>mc</td>
<td>A markup compatibility namespace that can be used to mark other namespaces/elements as ignorable. Normally used with the design-time namespace, whose attributes should be ignored at runtime.</td>
</tr>
</tbody>
</table>

The first two namespaces are almost always used in any XAML file. The second one (with the x prefix) is the **XAML language namespace**, which defines some special directives for the XAML parser. These directives often appear as attributes to XML elements, so they look like properties of the host element but actually are not. For a list of XAML keywords, see the “XAML Keywords” section later in this chapter.

---

**Most of the standalone XAML examples in this chapter explicitly specify their namespaces, but in the remainder of the book, most examples assume that the UI XML namespace (http://schemas.microsoft.com/winfx/2006/xaml/presentation) is declared as the primary namespace, and the XAML language namespace (http://schemas.microsoft.com/winfx/2006/xaml) is declared as a secondary namespace, with the prefix x.**
Using the UI XML namespace (http://schemas.microsoft.com/winfx/2006/xaml/presentation) as a default namespace and the XAML language namespace (http://schemas.microsoft.com/winfx/2006/xaml) as a secondary namespace with the prefix x is just a convention, just like it’s a convention to begin a C# file with a using System; directive. You could declare a Button in XAML as follows, and it would be equivalent to the Button defined previously:

```xml
```

Of course, for readability it makes sense for your most commonly used namespace (also known as the primary XML namespace) to be prefix free and to use short prefixes for any additional namespaces.

The last two namespaces in Table 2.1, which are plopped in pages generated by Visual Studio and Blend, are usually not needed.

---

**Markup Compatibility**

The markup compatibility XML namespace (http://schemas.openxmlformats.org/markup-compatibility/2006, typically used with an mc prefix) contains an Ignoorable attribute that instructs XAML processors to ignore all elements/attributes in specified namespaces if they can’t be resolved to their types/members. (The namespace also has a ProcessContent attribute that overrides Ignoorable for specific types inside the ignored namespaces.)

Blend and Visual Studio take advantage of this feature to do things like add design-time properties to XAML content that can be ignored at runtime. mc:Ignoorable can be given a space-delimited list of namespaces, and mc:ProcessContent can be given a space-delimited list of elements.

---

If you’re frustrated by how long it takes to open XAML files in Visual Studio and you don’t care about previewing the visuals, you might consider changing your default editor for XAML files by right-clicking on a XAML file in Solution Explorer then selecting Open With..., XML (Text) Editor, clicking Set as Default, then clicking OK. This has several major drawbacks, however, such as losing IntelliSense support.

---

**Property Elements**

Rich composition of controls is one of the highlights of XAML. This can be easily demonstrated with a Button, because you can put arbitrary content inside it; you’re not limited to just text! To demonstrate this, the following code embeds a simple square to make a Stop button like what might be found in a media player:

```csharp
b.Width = 96;
```
b.Height = 38;
r.Width = 10;
r.Height = 10;
r.Fill = new Windows.UI.Xaml.Media.SolidColorBrush(Windows.UI.Colors.White);
b.Content = r; // Make the square the content of the Button

Button's Content property is of type System.Object, so it can easily be set to the 10x10 Rectangle object. The result (when used with additional code that adds it to a page) is pictured in Figure 2.1.

That's pretty neat, but how can you do the same thing in XAML with property attribute syntax? What kind of string could you possibly set Content to that is equivalent to the preceding Rectangle declared in C#? There is no such string, but XAML fortunately provides an alternative (and more verbose) syntax for setting complex property values: property elements. It looks like the following:

```xml
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    Width="96" Height="38">
  <Button.Content>
    <Rectangle Width="10" Height="10" Fill="White"/>
  </Button.Content>
</Button>
```

The Content property is now set with an XML element instead of an XML attribute, making it equivalent to the previous C# code. The period in Button.Content is what distinguishes property elements from object elements. Property elements always take the form TypeName.PropertyName, they are always contained inside a TypeName object element, and they can never have attributes of their own (with one exception—the x:Uid attribute used for localization).

Property element syntax can be used for simple property values as well. The following Button that sets two properties with attributes (Content and Background):

```xml
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    Content="Stop" Background="Red"/>
```

is equivalent to this Button, which sets the same two properties with elements:

```xml
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
  <Button.Content>
    Stop
  </Button.Content>
  <Button.Background>
    Red
  </Button.Background>
</Button>
```
Of course, using attributes when you can is a nice shortcut when hand-typing XAML.

**Type Converters**

Let’s look at the C# code equivalent to the preceding `Button` declaration that sets both `Content` and `Background` properties:

```csharp
b.Content = "Stop";
b.Background = new Windows.UI.Xaml.Media.SolidColorBrush(Windows.UI.Color.Red);
```

Wait a minute. How can "Red" in the previous XAML file be equivalent to the `SolidColorBrush` instance used in the C# code? Indeed, this example exposes a subtlety with using strings to set properties in XAML that are a different data type than `System.String` or `System.Object`. In such cases, the XAML parser must look for a type converter that knows how to convert the string representation to the desired data type.

You cannot currently create your own type converters, but type converters already exist for many common data types. Unlike the XAML language, these type converters support case-insensitive strings. Without a type converter for `Brush` (the base class of `SolidColorBrush`), you would have to use property element syntax to set the `Background` in XAML as follows:

```xml
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    Content="Stop">
    <SolidColorBrush Color="Red"/>
</Button>
```

And even that is only possible because of a type converter for `Color` that can make sense of the "Red" string. If there wasn’t a `Color` type converter, you would basically be stuck. Type converters don’t just enhance the readability of XAML; they also enable values to be expressed that couldn’t otherwise be expressed.

Unlike in the previous C# code, in this case, misspelling `Red` would not cause a compilation error but would cause an exception at runtime. (Although Visual Studio does provide compile-time warnings for mistakes in XAML such as this.)
Markup Extensions

Markup extensions, like type converters, extend the expressiveness of XAML. Both can evaluate a string attribute value at runtime and produce an appropriate object based on the string. As with type converters, you cannot currently create your own, but several markup extensions are built in.

Unlike type converters, markup extensions are invoked from XAML with explicit and consistent syntax. Whenever an attribute value is enclosed in curly braces ({}), the XAML parser treats it as a markup extension value rather than a literal string or something that needs to be type-converted. The following Button uses two different markup extensions as the values for two different properties:

```xml
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       Height="50"
       Background="{x:Null}"
       Content="{Binding Height, RelativeSource={RelativeSource Self}}"/>
```

The first identifier in each set of curly braces is the name of the markup extension. The Null extension lives in the XAML language namespace, so the x prefix must be used. Binding (which also happens to be a class in the Windows.UI.Xaml.Data namespace), can be found in the default XML namespace. Note that the full name for Null is NullExtension, and this long form can be used as well in XAML. XAML permits dropping the Extension suffix on any markup extensions named with the suffix.

If a markup extension supports them, comma-delimited parameters can be specified. Positional parameters (such as Height in the example) are treated as string arguments for the extension class’s appropriate constructor. Named parameters (RelativeSource in the example) enable you to set properties with matching names on the constructed extension object. The values for these properties can be markup extension values themselves (using nested curly braces, as done with the value for RelativeSource) or literal values that can undergo the normal type conversion process. If you’re familiar with .NET custom attributes (the .NET Framework’s popular extensibility mechanism), you’ve probably noticed that the design and usage of markup extensions closely mirrors the design and usage of custom attributes. That is intentional.

In the preceding Button declaration, x:Null enables the Background brush to be set to null. This is just done for demonstration purposes, because a null Background is not very useful. Binding, covered in depth in Chapter 17, “Data Binding,” enables Content to be set to the same value as the Height property.
**Escaping the Curly Braces**

If you ever want a property attribute value to be set to a literal string beginning with an open curly brace ({}), you must escape it so it doesn't get treated as a markup extension. This can be done by preceding it with an empty pair of curly braces, as in the following example:

```xml
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
    <Button.Height>
        50
    </Button.Height>
    <Button.Background>
        <x:Null/>
    </Button.Background>
    <Button.Content>
        <Binding Path="Height">
            <Binding.RelativeSource>
                <RelativeSource Mode="Self"/>
            </Binding>
        </Binding>
    </Button.Content>
</Button>
```

Alternatively, you could use property element syntax without any escaping because the curly braces do not have special meaning in this context. The preceding `Button` could be rewritten as follows:

```xml
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
    <Button.Content>
        {This is not a markup extension!}
    </Button.Content>
</Button>
```

Markup extensions can also be used with property element syntax. The following `Button` is identical to the preceding one:

```xml
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
    <Button.Height>
        50
    </Button.Height>
    <Button.Background>
        <x:Null/>
    </Button.Background>
    <Button.Content>
        <Binding Path="Height">
            <Binding.RelativeSource>
                <RelativeSource Mode="Self"/>
            </Binding>
        </Binding>
    </Button.Content>
</Button>
```

This transformation works because these markup extensions all have properties corresponding to their parameterized constructor arguments (the positional parameters used with property attribute syntax). For example, `Binding` has a `Path` property that has the same meaning as the argument that was previously passed to its parameterized constructor, and `RelativeSource` has a `Mode` property that corresponds to its constructor argument.
Children of Object Elements

A XAML file, like all XML files, must have a single root object element. Therefore, it should come as no surprise that object elements can support child object elements (not just property elements, which aren’t children, as far as XAML is concerned). An object element can have three types of children: a value for a content property, collection items, or a value that can be type-converted to the object element.

The Content Property

Many classes designed to be used in XAML designate a property (via a custom attribute) that should be set to whatever content is inside the XML element. This property is called the content property, and it is just a convenient shortcut to make the XAML representation more compact.

Button’s Content property is (appropriately) given this special designation, so the following Button:

```xml
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
       Content="Stop"/>
```

could be rewritten as follows:

```xml
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
  Stop
</Button>
```

Or, more usefully, this Button with more complex content:

```xml
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
  <Button.Content>
    <Rectangle Height="10" Width="10" Fill="White"/>
  </Button.Content>
</Button>
```
could be rewritten as follows:

```xml
<Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
  <Rectangle Height="10" Width="10" Fill="White"/>
</Button>
```

There is no requirement that the content property must be called Content; classes such as ComboBox and ListBox (also in the Windows.UI.Xaml.Controls namespace) use their Items property as the content property.

### Collection Items

XAML enables you to add items to the two main types of collections that support indexing: lists and dictionaries.

#### Lists

A list is any collection that implements the `IList` interface or its generic counterpart. For example, the following XAML adds two items to a ListBox control whose Items property is an ItemCollection that implements `IList<object>`:

```xml
<ListBox xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
  <ListBox.Items>
    <ListBoxItem Content="Item 1"/>
    <ListBoxItem Content="Item 2"/>
  </ListBox.Items>
</ListBox>
```

This is equivalent to the following C# code:

```csharp
Windows.UI.Xaml.Controls.ListBox listbox = new Windows.UI.Xaml.Controls.ListBox();
Windows.UI.Xaml.Controls.ListBoxItem item1 = new Windows.UI.Xaml.Controls.ListBoxItem();
Windows.UI.Xaml.Controls.ListBoxItem item2 = new Windows.UI.Xaml.Controls.ListBoxItem();
item1.Content = "Item 1";
item2.Content = "Item 2";
listbox.Items.Add(item1);
listbox.Items.Add(item2);
```

Furthermore, because Items is the content property for ListBox, you can shorten the XAML even further, as follows:

```xml
<ListBox xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
  <ListBoxItem Content="Item 1"/>
</ListBox>
```
In all these cases, the code works because ListBox’s Items property is automatically initialized to any empty collection object. If a collection property is initially null instead (and is read/write, unlike ListBox’s read-only Items property), you would need to wrap the items in an explicit element that instantiates the collection. The built-in controls do not act this way, so an imaginary OtherListBox element demonstrates what this could look like:

```
<OtherListBox>
<OtherListBox.Items>
<ItemCollection>
<ListBoxItem Content="Item 1"/>
<ListBoxItem Content="Item 2"/>
</ItemCollection>
</OtherListBox.Items>
</OtherListBox>
```

**Dictionaries**

A dictionary is any collection that implements the IDictionary interface or its generic counterpart. Windows.UI.Xaml.ResourceDictionary is a commonly used collection type that you’ll see more of in later chapters. It implements IDictionary<object, object>, so it supports adding, removing, and enumerating key/value pairs in procedural code, as you would do with a typical hash table. In XAML, you can add key/value pairs to any dictionary. For example, the following XAML adds two Colors to a ResourceDictionary:

```
<ResourceDictionary
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
 <Color x:Key="1">White</Color>
 <Color x:Key="2">Black</Color>
</ResourceDictionary>
```

This leverages the XAML Key keyword (defined in the secondary XML namespace), which is processed specially and enables us to attach a key to each Color value. (The Color type does not define a Key property.) Therefore, the XAML is equivalent to the following C# code:

```csharp
Windows.UI.Color color1 = Windows.UI.Colors.White;
Windows.UI.Color color2 = Windows.UI.Colors.Black;
d.Add("1", color1);
d.Add("2", color2);
```

Note that the value specified in XAML with x:Key is treated as a string unless a markup extension is used; no type conversion is attempted otherwise.
More Type Conversion
Plain text can often be used as the child of an object element, as in the following XAML declaration of SolidColorBrush:

```xml
<SolidColorBrush>White</SolidColorBrush>
```

This is equivalent to the following:

```xml
<SolidColorBrush Color="White"/>
```

even though Color has not been designated as a content property. In this case, the first XAML snippet works because a type converter exists that can convert strings such as "White" (or "white" or "#FFFFFF") into a SolidColorBrush object.

Although type converters play a huge role in making XAML readable, the downside is that they can make XAML appear a bit “magical,” and it can be difficult to understand how it maps to instances of objects. Using what you know so far, it would be reasonable to assume that you can’t declare an instance of a class in XAML if it has no default constructor. However, even though the Windows.UI.Xaml.Media.Brush base class for SolidColorBrush, LinearGradientBrush, and other brushes has no constructors at all, you can express the preceding XAML snippets as follows:

```xml
<Brush>White</Brush>
```

The type converter for Brushes understands that this is still SolidColorBrush. This might seem like an unusual feature, but it’s important for supporting the ability to express primitive types in XAML, as demonstrated in “The Extensible Part of XAML.”

---

The Extensible Part of XAML

Because XAML was designed to work with the .NET type system, you can use it with just about any object, including ones you define yourself. It doesn't matter whether these objects have anything to do with a user interface. However, the objects need to be designed in a “declarative-friendly” way. For example, if a class doesn't have a default constructor and doesn't expose useful instance properties, it’s not going to be directly usable from XAML. A lot of care went into the design of the APIs in the Windows.UI.Xaml namespace—above and beyond the usual design guidelines—to fit XAML’s declarative model.

To use an arbitrary .NET class (with a default constructor) in XAML, simply include the proper namespace with using syntax. The following XAML does this with an instance of System.Net.Http.HttpClient and System.Int64:

```xml
<ListBox xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:sys="using:System">
  <ListBox.Items>
    <sys:Int64 xmlns:sys="using:System">100</sys:Int64>
  </ListBox.Items>
</ListBox>
```
Mixing XAML with Procedural Code

XAML-based Windows Store apps are a mix of XAML and procedural code. This section covers the two ways that XAML and code can be mixed together: dynamically loading and parsing XAML yourself, or leveraging the built-in support in Visual Studio projects.

Loading and Parsing XAML at Runtime

The Windows.UI.Xaml.Markup namespace contains a simple XamlReader class with a simple static Load method. Load can parse a string containing XAML, create the appropriate Windows Runtime objects, and return an instance of the root element. So, with a string containing XAML content somewhat like MainPage.xaml from the preceding chapter, the following code could be used to load and retrieve the root Page object:

```csharp
string xamlString = ...;
// Get the root element, which we know is a Page
Page p = (Page)XamlReader.Load(xamlString);
```

After Load returns, the entire hierarchy of objects in the XAML file is instantiated in memory, so the XAML itself is no longer needed. Now that an instance of the root element exists, you can retrieve child elements by making use of the appropriate content

The XAML language namespace defines keywords for a few common primitives so you don’t need to separately include the System namespace: x:Boolean, x:Int32, x:Double, and x:String.
properties or collection properties. The following code assumes that the Page has a StackPanel object as its content, whose fifth child is a Stop button:

```csharp
string xamlString = ...;
// Get the root element, which we know is a Page
Page p = (Page)XamlReader.Load(xamlString);
// Grab the Stop button by walking the children (with hard-coded knowledge!)
StackPanel panel = (StackPanel)p.Content;
Button stopButton = (Button)panel.Children[4];
```

With a reference to the Button control, you can do whatever you want: Set additional properties (perhaps using logic that is hard or impossible to express in XAML), attach event handlers, or perform additional actions that you can’t do from XAML, such as calling its methods.

Of course, the code that uses a hard-coded index and other assumptions about the user interface structure isn’t satisfying, because simple changes to the XAML can break it. Instead, you could write code to process the elements more generically and look for a Button element whose content is a "Stop" string, but that would be a lot of work for such a simple task. In addition, if you want the Button to contain graphical content, how can you easily identify it in the presence of multiple Buttons?

Fortunately, XAML supports naming of elements so they can be found and used reliably from procedural code.

**Naming XAML Elements**

The XAML language namespace has a Name keyword that enables you to give any element a name. For the simple Stop button that we’re imagining is embedded somewhere inside a Page, the Name keyword can be used as follows:

```xml
<Button x:Name="stopButton">Stop</Button>
```

With this in place, you can update the preceding C# code to use Page’s FindName method that searches its children (recursively) and returns the desired instance:

```csharp
string xamlString = ...;
// Get the root element, which we know is a Page
Page p = (Page)XamlReader.Load(xamlString);
// Grab the Stop button, knowing only its name
Button stopButton = (Button)p.FindName("stopButton");
```

FindName is not unique to Page; it is defined on FrameworkElement, a base class for many important classes in the XAML UI Framework.
Naming Elements Without \texttt{x:Name}

The \texttt{x:Name} syntax can be used to name elements, but \texttt{FrameworkElement} also has a name property that accomplishes the same thing. You can use either mechanism on such elements, but you can't use both simultaneously. Having two ways to set a name is a bit confusing, but it's handy for these classes to have a name property for use by procedural code. Sometimes you want to name an element that doesn't derive from \texttt{FrameworkElement} (and doesn't have a name property), so \texttt{x:Name} is necessary for such cases.

Visual Studio's Support for XAML and Code-Behind

Loading and parsing XAML at runtime can be interesting for some limited dynamic scenarios. Windows Store projects, however, leverage work done by MSBuild and Visual Studio to make the combination of XAML and procedural code more seamless. When you compile a project with XAML files, the XAML is included as a resource in the app being built and the plumbing that connects XAML with procedural code is generated automatically.

The automatic connection between a XAML file and a code-behind file is enabled by the \texttt{Class} keyword from the XAML language namespace, as seen in the preceding chapter. For example, \texttt{MainPage.xaml} had the following:

\begin{verbatim}
(Page \texttt{x:Class="BlankApp.MainPage"} ...)
...
</Page>
\end{verbatim}

This causes the XAML content to be treated as a partial class definition for a class called \texttt{MainPage} (in the \texttt{BlankApp} namespace) derived from \texttt{Page}. The other pieces of the partial class definition reside in auto-generated files as well as the \texttt{MainPage.xaml.cs} code-behind file. Visual Studio's Solution Explorer ties these two files together by making the code-behind file a subnode of the XAML file, but that is an optional cosmetic effect enabled by the following XML inside of the .csproj project file:

\begin{verbatim}
<Compile Include="MainPage.xaml.cs">
  <DependentUpon>MainPage.xaml</DependentUpon>
</Compile>
\end{verbatim}

You can freely add members to the class in the code-behind file. And if you reference any event handlers in XAML (via event attributes such as \texttt{Click} on \texttt{Button}), this is where they should be defined.

Whenever you add a page to a Visual Studio project (via Add New Item...), Visual Studio automatically creates a XAML file with \texttt{x:Class} on its root, creates the code-behind source file with the partial class definition, and links the two together so they are built properly.
The additional auto-generated files alluded to earlier contain some “glue code” that you normally never see and you should never directly edit. For a XAML file named `MainPage.xaml`, they are:

➔ `MainPage.g.cs`, which contains code that attaches event handlers to events for each event attribute assigned in the XAML file.

➔ `MainPage.g.i.cs`, which contains a field definition (private by default) for each named element in the XAML file, using the element name as the field name. It also contains an `InitializeComponent` method that the root class’s constructor must call in the code-behind file. This file is meant to be helpful to IntelliSense, which is why it has an “i” in its name.

The “g” in both filenames stands for `generated`. Both generated source files contain a partial class definition for the same class partially defined by the XAML file and code-behind file.

If you peek at the implementation of `InitializeComponent` inside the auto-generated file, you’ll see that the hookup between C# and XAML isn’t so magical after all. It looks a lot like the code shown previously for manually loading XAML content and grabbing named elements from the tree of instantiated objects. Here’s what the method looks like for the preceding chapter’s `MainPage` if a `Button` named `stopButton` were added to it:

```csharp
public void InitializeComponent()
{
    if (_contentLoaded)
        return;

    _contentLoaded = true;
    Application.LoadComponent(this, new System.Uri("ms-appx:///MainPage.xaml"),
                              Windows.UI.Xaml.Controls.Primitives.ComponentResourceLocation.Application);

    stopButton = (Windows.UI.Xaml.Controls.Button)this.FindName("stopButton");
}
```

The `LoadComponent` method is much like `XamlReader`'s `Load` method, except it works with a reference to an app’s resource file.

To reference a resource file included with your app, simply use a URI with the format "ms-appx:///relative path to file". XAML files are already treated specially, but adding a new resource file to your app is as simple as adding a new file to your project with a `Build Action` of `Content`. Chapter 11, “Images,” shows how to use resources such as image files with the `Image` element.
XAML Keywords

The XAML language namespace (http://schemas.microsoft.com/winfx/2006/xaml) defines a handful of keywords that must be treated specially by any XAML parser. They mostly control aspects of how elements get exposed to procedural code, but several are useful independent of procedural code. You've already seen some of them (such as Key, Name, and Class), but Table 2.2 lists all the ones relevant for Windows Store apps. They are listed with the conventional x prefix because that is how they usually appear in XAML and in documentation.

### Table 2.2 Keywords in the XAML Language Namespace, Assuming the Conventional x Namespace Prefix

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Valid As</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>x:Class</td>
<td>Attribute on root element.</td>
<td>Defines a namespace-qualified class for the root element that derives from the element type.</td>
</tr>
<tr>
<td>x:FieldModifier</td>
<td>Attribute on any nonroot element but must be used with x:Name (or equivalent).</td>
<td>Defines the visibility of the field to be generated for the element (which is private by default). The value must be specified in terms of the procedural language (for example, public, private, and internal for C#).</td>
</tr>
<tr>
<td>x:Key</td>
<td>Attribute on an element whose parent is a dictionary.</td>
<td>Specifies the key for the item when added to the parent dictionary.</td>
</tr>
<tr>
<td>x:Name</td>
<td>Attribute on any nonroot element but must be used with x:Class on root.</td>
<td>Chooses a name for the field to be generated for the element, so it can be referenced from procedural code.</td>
</tr>
<tr>
<td>x:Null</td>
<td>An element or an attribute value as a markup extension. Can also appear as x:NullExtension.</td>
<td>Represents a null value.</td>
</tr>
<tr>
<td>x:StaticResource</td>
<td>An element or an attribute value as a markup extension. Can also appear as x:StaticResourceExtension.</td>
<td>References a XAML resource</td>
</tr>
</tbody>
</table>

Special Attributes Defined by the W3C

In addition to keywords in the XAML language namespace, XAML also supports two special attributes defined for XML by the World Wide Web Consortium (W3C): xml:space for controlling whitespace parsing and xml:lang for declaring the document’s language and culture. The xml prefix is implicitly mapped to the standard XML namespace; see http://www.w3.org/XML/1998/namespace.
### Summary

You have now seen how XAML fits in with the rest of an app’s code and, most importantly, you now have the information needed to translate most XAML examples into a language such as C# and vice versa. However, because type converters and markup extensions are “black boxes,” a straightforward translation is not always going to be obvious.

As you proceed further, you might find that some APIs can be a little clunky from procedural code because their design is often optimized for XAML use. For example, the XAML UI Framework exposes many small building blocks to help enable rich composition, so some scenarios can involve manually creating a lot of objects. Besides the fact the XAML excels at expressing deep hierarchies of objects concisely, Microsoft spent more time implementing features to effectively hide intermediate objects in XAML (such as type converters) rather than features to hide them from procedural code (such as constructors that create inner objects on your behalf).

Most people understand the benefit of XAML’s declarative model, but some lament XML as the choice of format. The primary complaint is that it’s verbose; too verbose to type. This is true: Almost nobody enjoys typing lots of XML, but that’s where tools come in. Tools such as IntelliSense and visual designers can spare you from typing a single angle bracket! The transparent and well-specified nature of XML enables you to easily integrate new tools into the development process (creating a XAML exporter for your favorite tool, for example) and also enables easy hand-tweaking or troubleshooting.

In some areas (such as complicated paths and shapes), typing XAML by hand isn’t even practical. In fact, the trend from when XAML was first introduced in beta form has been to remove some of the handy human-typeable shortcuts in favor of a more robust and extensible format that can be supported well by tools. But I still believe that being familiar with XAML and seeing the APIs through both procedural and declarative perspectives is the best way to learn the technology. It’s like understanding how HTML works without relying on a visual tool.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Valid As</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>x:Subclass</td>
<td>Attribute on root element and must be used with x:Class.</td>
<td>Specifies a subclass of the x:Class class that holds the content defined in XAML. This is needed only for languages without support for partial classes, so there's no reason to use this in a C# XAML project.</td>
</tr>
<tr>
<td>x:TemplateBinding</td>
<td>An element or an attribute value as a markup extension.</td>
<td>Binds to an element's properties from within a template, as described in Chapter 16. Can also appear as x:TemplateBindingExtension.</td>
</tr>
<tr>
<td>x:Uid</td>
<td>Attribute on any element</td>
<td>Marks an element with an identifier used for localization.</td>
</tr>
</tbody>
</table>
Classes in the XAML UI Framework have a deep inheritance hierarchy, so it can be hard to get your head wrapped around the significance of various classes and their relationships. A handful of fundamental classes are referenced often and deserve a quick explanation before we get any further in the book. The Page class, for example, derives from a UserControl class, which derives from all of the following classes, in order from most to least derived:

- **Control**—The base class for familiar controls such as Button and ListBox. Control adds many properties to its base class, such as Foreground, Background, and FontSize, as well as the capability to be given a completely new visual template. Part III, “Understanding Controls,” examines the built-in controls in depth.

- **FrameworkElement**—The base class that adds support for styles, data binding, XAML resources, and a few common mechanisms such as tooltips and context menus.

- **UIElement**—The base class for all visual objects with support for routed events, layout, and focus. These features are discussed in Chapter 4, “Layout,” and Chapter 5, “Interactivity.”

- **DependencyObject**—The base class for any object that can support dependency properties, also discussed in Chapter 5.

- **Object**—The base class for all .NET classes.

Throughout the book, the simple term *element* is used to refer to an object that derives from UIElement or FrameworkElement. The distinction between UIElement and FrameworkElement is not important because the framework doesn’t include any other public subclasses of UIElement.
Symbols and Numbers

&#xHexValue, 202

2D transforms, 55-56
- CompositeTransform, 60
- MatrixTransform, 61-62
- RotateTransform, 56-57
- ScaleTransform, 57-59
- SkewTransform, 59-60
- TranslateTransform, 60
- TransformGroup, 61

3D transforms, 62, 64

A

absolute sizing, 80
absolute URIs, 256
AccelerationX property (AccelerometerReading class), 530
AccelerationY property (AccelerometerReading class), 530
AccelerationZ property (AccelerometerReading class), 530, 532
AcceleratorKeyActivated event, 143
accelerometer, 529-531
- shake detection, 532
- tossing motion, 531-532
Accelerometer.GetDefault method, 530
AccelerometerReading class, properties, 530
AcceptsReturn, TextWrapping versus, 241
account pictures
extensions, 509-511
providers, 510
AccountPictureChanged event, 509
Accuracy property (Coordinate property), 534
Activate event (Windows class), 156
Activated event (Window.Current property), 153
activating (action), 155-156, 159-160
ActivationKind property (OnLaunched), 157
ActualHeight property, 49
actualWidth property, 49
AddDeleteThemeTransition, 372-374
AddHandler method, 112
addPages event, 495
affine transformation matrix, 61
AgeRating property (ListingInformation object), 169
alignment, child layout properties
content alignment, 53-55
HorizontalAlignment and VerticalAlignment, 52-53
AllowCropping property (PhotoSettings property), 296
AllowDrop property, 140
AllowTrimming property (VideoSettings property), 298
Alt key, 143
altform-XXX resource qualifier, 266
Altitude property (Coordinate property), 534
AltitudeAccuracy property (Coordinate property), 535
ambient light sensor, 533
Angle property, 56
animation
custom animations, 382-383
ColorAnimation class, 387, 391-393
data types, 382
dependency properties, 383
DiscreteXXXKeyFrame class, 398-399
DoubleAnimation class, 382-385
DoubleAnimationUsingKeyFrames class, 395-397
duration, 384-385
EasingXXXKeyFrame class, 399
From property, 385-387
independent versus dependent, 383-384
ITranslateTransform, 384
KeySpline class, 397
PointAnimation class, 387, 382
property paths, 391-393
ScaleTransform, 384
Timeline class properties, 388-390, 393, 395
to property, 385-387
dependent, 366
EasingFunction property, 400
power easing, 400-401
independent, 366
keyframes, 395
discrete, 398-399
easing, 399
linear, 395-396
spline, 397
manual, 404-406
theme animations, 366, 376
DragItemThemeAnimation, 380
DragOverThemeAnimation, 380
DropTargetItemThemeAnimation, 381
Duration property, 381
FadeInThemeAnimation, 379, 381
FadeOutThemeAnimation, 376-381
multiple storyboards, 390-391
PointerDownThemeAnimation, 380
PointerUpThemeAnimation, 380
PopInThemeAnimation, 379
PopOutThemeAnimation, 380
RepositionThemeAnimation, 380
SplitCloseThemeAnimation, 381
SplitOpenThemeAnimation, 381
storyboards, 376-379, 390-391
SwipeBackThemeAnimation, 380
SwipeHintThemeAnimation, 380
Timeline class properties, 381, 388-390
theme transitions, 366
AddDeleteThemeTransition, 372-374
applying to elements, 366-367
ContentThemeTransition, 370
EdgeUIThemeTransition, 370-371
EntranceThemeTransition, 368-369
PaneThemeTransition, 371-372
PopupThemeTransition, 369-370
ReorderThemeTransition, 375-376
RepositionThemeTransition, 374-375
TransitionCollection element, 367

App class, 21
app container, 9

App data, 461
app files, 464
  local files, 465
  packaged files, 464
  roaming files, 465-466
temporary files, 466
app settings, 462
  local settings, 462
  roaming settings, 463-464

app files, App data, 464
  local files, 465
  packaged files, 464
  roaming files, 465-466
temporary files, 466

app lifecycle, 149
  execution states, 150-151
  transition actions, 150-151

activating, 155-156, 159-160
killing, 152
launching, 155-159
managing session state with SuspensionManager class, 160-163
resuming, 154
suspending, 152-154
terminating, 155

app settings, 462
  local settings, 462
  roaming settings, 463-464

App.xaml.cs, 22, 24

AppBar control, 132, 196-197
  AppBarButtonStyle, 200-204
  attaching to pages, 197-198
  buttons, 199-202, 204-205
design guidelines, 198-200
  IsSticky property, 198
  pressed appearances, 205

AppBarButtonStyle (AppBar control), 200-205

AppBars (ListView control), 218

Application class, 159-160
  app transitions, 150
  Exit method, 152
  Suspending event, 152-153
deferrals, 154
  handling, 153-154

application definition, 21
  App.xaml.cs, 22, 24
  AssemblyInfo.cs, 24

application state (files), 254

Application UI tab (package manifest), 10
  splash screen customization, 10-11
tile customization, 12-13

ApplicationData.SetVersionAsync method, 462
ApplicationDataCompositeValue class, 463
apps

colors, 190
.crashes, 152
data. See app data
files. See app files
launching programmatically, 163
customizing app launches, 165-166
for files, 163-164
for URIs, 164-165
layout, 47
controlling position, 52-55
controlling size, 48-51
perspective transforms, 62, 64
transform classes, 55-62
lifecycle. See app lifecycle
models, 149, 166
navigation, 174
Back button, 179
back navigation, 176-177
basic navigation, 175-176
embedding frames, 180-182
forward navigation, 176-177
LayoutAwarePage class, 178-179
page caching, 177-178
passing data, 175-176
orientation, 10
pages, 174
back navigation, 176-177
basic navigation, 175-176
forward navigation, 176-177
passing data, 175-176
Search pane, 478
settings. See app settings
themes, 186
tombstoning, 183
Windows Store, 7
Blank App project, 8-14, 16-19, 21-22, 24
ArcSegment class (PathSegments), 341
Arguments property (OnLaunched), 156
AssemblyInfo.cs, 24
AtomPubClient class, 474
attached properties, 71-73, 107-108
attached property provider, 72
attributes, 28-29
audio, 285-286
background tasks, 520
adding declaration, 520
AudioCategory property, 520
media transport controls, 521-523
capture, 294
CameraCaptureUI class, 294
CaptureElement class, 304-305
microphones, 294
formats, MediaElement, 286-287
metadata, 289
playback, 286
MediaElement, 286-291
MediaPlayer, 291-292, 294
transcoding (MediaTranscoder class), 305-310
AudioCategory property, 520
AudioDeviceController property
(MediaCapture), 305
AudioDeviceld property (InitializeAsync overload), 299
Auto state (ScrollBarVisibility), 91
automatic scaling, image files, 263
loading file variations automatically, 263-266
loading file variations manually, 266-267
AutomationProperties.Name property, 201
automaton IDs, items controls, 215
AutoPlay content, extensions, 512-514
AutoPlay device, extensions, 512-514
AutoReverse property (Timeline class), 388
AutoRotationPreferences property, 70
autosizing, 80
B

Back button, 179
CoreWindowDialog, 319
Back navigation, 176-177
BackEase function, 403
background color, tile, 13
background downloads, network data access, 471
Background property, 124
background tasks
audio, 520
adding declaration, 520
AudioCategory property, 520
media transport controls, 521-523
customizing
applying conditions, 527
event progress, 525
IbackgroundTask implementation, 523-524
registering task, 524-525
triggers, 525-526
background video, 291
BackgroundDownloader class, 471
badges, live tiles, 549-550
BaseUri property (Page), 255
BasicProperties class, 272
BeginTime property (Timeline class), 388
Bézier curves, 341
control points, 341
S-like shapes, 341
U-like shapes, 341
BezierSegment class (PathSegments), 341
BGRA8 pixel format, 261
binary data, 462
binding, 439-440
C#, 441
to collections, 444-446
Convert method, 454
Converter property, 454
customizing data flow, 442-443
data templates, 448
Mode property (Binding object), 442
OneWay binding, 443
to plain properties, 442
RelativeSource property (Binding object), 441
sharing source with DataContext, 443-444
source property (Binding object), 440
target property (Binding object), 440
TwoWay binding, 443
Binding markup extension
binding to collections, 444-446
binding to plain properties, 442
C#, 441
customizing data flow, 442-443
data templates, 448
Mode property, 442
OneWay binding, 443
RelativeSource property, 441
sharing source with DataContext, 443-444
source property, 440
target property, 440
TwoWay binding, 443
Binding object, 439-440
BindingMode enumeration, 442
BitmapDecoder, reading metadata with BitmapProperties, 273-275
BitmapDecoder class, decoding images, 267
going pixel data, 268-269, 271
BitmapEncoder class, 276-277
writing metadata, 279-280
writing pixel data, 277-279
BitmapImage, 255
BitmapProperties property (Bitmap Decoder), reading image metadata, 273-276
BitmapTransform class, 270-271

Blank App, 7
  launching a new app, 8-9
    application definition, 21-22, 24
    Main Page, 19, 21
    package manifest, 9-14, 16-19

Block (TextElements), 232

Border, ChildTransitions property, 367

BottomAppBar property, 197-198

BounceEase function, 403

Brush data type, 348
  ImageBrush, 355-356
  LinearGradientBrush, 349-355
  SolidColorBrush, 348-349
  WebViewBrush, 358-363

brushes, 348
  ImageBrush, 355-356
  LinearGradientBrush, 349-355
  SolidColorBrush, 348-349
  WebViewBrush, 358-363

bubbling event, 108, 110-111
  halting, 111-113

built-in libraries, 16

business models, 166

Button control, 28, 188-189
  default appearance, 185

Button states, 429, 432

ButtonAutomationPeer class, 189

buttons, 188-189
  AppBar control, 199-205
  AppBarButtonStyle, 201-204
  Back, 179
  MediaPlayer, 293
  Start Debugging, 8

Button_Click method (C#), 29

Byte properties (Color Property), 348

C

C#
  data binding, 441

Windows Store apps, 7
  Blank App project, 8-14, 16-19, 21-22, 24

cache, session states, 163

cached composition, 364

caching pages, 177-178

CameraCaptureUI class, 294
  photo captures, 294-297
  PhotoSettings property, 296
  video captures, 297-298
  VideoSettings property, 298

Cameras
  adjusting settings, 302-303
  live content, 299

Canceled event (EdgeGesture), 132

CanDragItems property, 140

CanExecute method, 114

CanExecuteChanged method, 114

CanPlayType method (MediaElement), 287

Canvas panel, 71-73
  Grid panel mimicking, 82-83

Capabilities tab (package manifest), 14-15
  device capabilities, 16-17
  file capabilities, 16
  identity capabilities, 17
  network capabilities, 17

capture, 294
  CameraCaptureUI class, 294
    photo captures, 294-297
    video captures, 297-298
  CaptureElement class, 298
    adjusting camera settings, 302-303
    capturing audio, 304-305
capturing photos, 301-302
capturing video, 303-304
showing live previews, 298-301
microphones, 294
Webcams, 294
capture and release pointers, 120, 122
CaptureElement class, 298
  adjusting camera settings, 302-303
capturing audio, 304-305
capturing photos, 301-302
capturing video, 303-304
showing live previews, 298-301
Source property, 298
Stretch property, 298
CaptureFileAsync, 296
CapturePhotoToStorageFileAsync method (MediaCapture), 302
CapturePointer method, 120
CarouselPanel, 210-212
CenterOfRotationX property, 64
CenterOfRotationY property, 64
CenterOfRotationZ property, 64
CenterX property, 56
CenterY property, 56
change notification, dependency properties, 104
Character Map, 202
CharacterSpacing property (TextBox), 228-229
chars, 477
  Devices, 492-493
    Play To feature, 501-502
    printing, 493-500
Search, 477-479
  auto-generated search results Page, 479
  reporting search results, 479-482
  search suggestions, 483-485
  search suggestions from indexed files, 485-486
Settings, 503-507
Share, 486
  share sources, 486-489
  share targets, 489-490, 492
charms bar, 477
CheckBox control, 192
CheckFeatureLicenses method, 170
child elements
  layout properties, 48, 52
    alignment, 52-53
    content alignment, 53-55
    Height and Width, 48-49
    Margin and Padding, 50-51
  perspective transforms, 62, 64
  transform classes, 55-56
    CompositeTransform, 60
    MatrixTransform, 61-62
    RotateTransform, 56-57
    ScaleTransform, 57-59
    SkewTransform, 59-60
    TranslateTransform, 60
    TransformGroup, 61
child layout properties
  Canvas panel, 72
  Grid panel, 79
  StackPanel panel, 74
  VariableSizedWrapGrid panel, 86
child object elements, 36
  collection items, 37
    dictionaries, 38
    lists, 37-38
  content property, 36-37
    type-converted values, 39-40
ChildrenTransitions property, 367
ChildTransitions property, 367
chips, NFC, 535
CircleEase function, 403
CivicAddress property, 534-535
classes. See also subclasses

AccelerometerReading properties, 530
Application, 159-160
  app transitions, 150
  Exit, 152
  Suspending event, 152-154
ApplicationDataCompositeValue, 463
ArcSegment (PathSegments), 341
AtomPubClient, 474
BackgroundDownloader, 471
BezierSegment (PathSegments), 341
ButtonAutomationPeer, 189
CameraCaptureUI, 294
  photo captures, 294-297
  PhotoSettings property, 296
  video captures, 297-298
CameraCaptureUI, VideoSettings property, 298
CaptureElement, 298
  adjusting camera settings, 302-303
  capturing audio, 304-305
  capturing photos, 301-302
  capturing video, 303-304
  showing live previews, 298-301
  Source property, 298
  Stretch property, 298
ColorAnimation, 387, 391-393
Compass, 533
ContentControl, 187
Control, 53
CurrentAppSimulator, testing Windows Store features, 172, 174
DiscreteXXXKeyFrame, 398-399
DispatcherTimer, 404
DoubleAnimation, 382-385
DoubleAnimationUsingKeyFrames, 395-397
EasingXXXKeyFrame, 399
Ellipse, 335
EllipseGeometry (Geometry data type), 340
Geolocator class, 534-535
GeometryGroup (Geometry data type), 340, 344
  FillRule property, 344
  strings, 346-348
  triangles, 345
Gyrometer, 532
ItemsControl, 207
  Items property, 208-209
  ItemsSource property, 209
  Selector subclass, 207-208
KeySpline, 397
Launcher
  LaunchFileAsync method, 163-166
  LaunchUriAsync method, 164-166
LayoutAwarePage, 163, 178-179, 432
Line, 336
LineGeometry (Geometry data type), 340
LineSegment (PathSegments), 341-342
MediaEncodingProfile, 304, 308
MediaEncodingProfile class, 306
MediaTranscoder, 305
  adding effects, 310
  changing media format, 308
  changing quality, 306-308
  PrepareStreamTranscodeAsync method, 306
  trimming files, 309-310
ObservableCollection, 445
Panel, 47
Path, 338
PathGeometry (Geometry data type), 340-341
  FillRule property, 343-344
  PathFigures, 341, 343
  PathSegments, 341-342
Playlist, 293
PointAnimation, 382, 387
PolyBezierSegment (PathSegments), 341
Polygon, 337
Polyline, 336-337
PolyLineSegment (PathSegments), 341
PolyQuadraticBezierSegment (PathSegments), 341
PrintDocument, 493, 495-496, 498
PrintManager, 493
QuadraticBezierSegment (PathSegments), 341
RadialGradientBrush, 355
RangeBase, 313
Rectangle, 334-335
  RadiusX property, 334
  RadiusY property, 334
RectangleGeometry (Geometry data type), 340
Shape
  overuse of shapes, 340
  StrokeDashArray property, 338-339
  StrokeDashCap property, 338-339
  StrokeEndLineCap property, 338-339
  StrokeLineJoin property, 338
  StrokeMiterLimit property, 338
  StrokeStartLineCap property, 338-339
  StrokeThickness property, 338
SplineXXXKeyFrame, 397
StandardDataFormats, 489
StorageFile, data binding, 449
SuspensionManager
  ISessionState property, 162
  managing session state, 160-163
  RestoreAsync method, 163
  SaveAsync method, 163
Thickness, 50
ThreadPoolTimer, 404
TileBrush, 355
ToolTipService class (ToolTip control), 194-196
VisualState, 429
Windows
  Activate event, 156
  VisibilityChanged event, 156
Windows.Storage.ApplicationData, 461
Windows.System.Launcher, 163
  customizing app launches, 165-166
  launching apps for files, 163-164
  launching apps for URIs, 164-165
Click event, 188-189
  RepeatButton control, 191
ClickMode property, 188
clipping content overflow, 87-89
collection items, 37
  dictionaries, 38
  lists, 37-38
collections
  binding to, 444-446
  Items properties, 446
  Markers collection, 290
  views, 455
    groupings, 455-459
    navigating, 459
color brushes
  LinearGradientBrush, 349-355
  SolidColorBrush, 348-349
color strings, 348
ColorAnimation class, 387, 391-393
ColorInterpolationMode property (LinearGradientBrush), 350
colors
  apps, 190
  gradients, 391-393
  themes, 190
  translucency, 349
ColumnDefinitions property (Grid panel), 76
columns, sizing, 79-82
ColumnSpan attached property, 76, 78, 211
    VariableSizedWrapGrid panel, 84-85
ComboBox control, 213
    DropDownClosed event, 213
    DropDownOpened event, 213
    IsDropDownOpen property, 213
    IsEditable property, 214
    keystrokes, 214
    SelectionChanged event, 214
commands, 113-114
    custom, CoreWindowDialog, 318
    Geometry string, 346-347
    header-embedded script, 290
    separate-stream script, 290
commas, Geometry strings, 348
communication, peer devices, 537-538
Compass class, 533
compass sensor, 533
CompassReading type, properties, 533
Completed event (EdgeGesture), 132
CompositeTransform class, 60
ComputedHorizontalScrollBarVisibility property (ScrollViewer), 92
ComputedVerticalScrollBarVisibility property (ScrollViewer), 92
config-XXX resource qualifier, 266
connected standby mode, 153
constructors, App.xaml.cs, 22
ContactRect property (PointerPointProperties class), 118
ContactRectRaw property (PointerPointProperties class), 118
contacts, Windows contact picker, 514-516
containers (item), 209
content
    alignment, child layout properties, 53-55
    AutoPlay, 512-514
text
    TextBlock, 229-231
    TextElements, 232-233
content control template
    Content property, 420-422
    hijacking existing properties, 426
    honoring properties, 423-425
content controls, 187
    AppBar, 196-197
    AppBarButtonStyle, 200-204
    attaching to pages, 197-198
    design guidelines, 198-200
    pressed appearances, 205
    Button, 188-189
    CheckBox, 192
    ContentTransitions property, 367
    HyperlinkButton, 189-190
    objects, 187
    RadioButton, 192-193
    RepeatButton, 191
    ToggleButton, 191
    ToolTip, 194-196
content overflow, 87
    clipping, 87-89
    scaling, 94
        Viewbox element, 95-98
        ZoomMode property, 98
    scrolling, 89-90
        customizing ScrollViewer, 90-91, 93
        snap points, 93-94
Content property, 36-37
    content control template, 420-422
    Frame, 177
    MessageDialog popup control, 322
    ScrollViewer, 89
ContentControl class, 187, 447
ContentEnd property (TextBlock), 234
ContentStart property (TextBlock), 234
ContentTemplate property (ContentControl), 447
ContentThemeTransition, 370
ContentTransitions property, 367
ContextMenuOpening event, 234
contracts, 18
contrast resource qualifiers, 265
contrast-black qualifiers, 265
contrast-high qualifiers, 265
contrast-standard qualifiers, 265
contrast-white qualifiers, 265
Control class, 53
control points, Bézier curves, 341
control templates, XAML control restyling, 419-420
controls
  AppBar
    buttons, 199-205
    IsSticky property, 198
  Button, 28
    default appearance, 185
content, 187
  AppBar, 196-205
  Button, 188-189
  CheckBox, 192
  ContentTransitions property, 367
  HyperlinkButton, 189-190
  objects, 187
  RadioButton, 192-193
  RepeatButton, 191
  ToggleButton, 191
  ToolTip, 194-196
  CustomSettingsPane, 505-507
dark theme, 185
items, 207
    automaton IDs, 215
    ComboBox, 213-214
data binding, 209
FlipView, 221-223
GridView, 219-221
ItemContainerTransitions property, 367
items panels, 210-212
ItemsControl class, 208-209
ListBox, 214-215
ListView, 216-219
SemanticZoom, 223-226
visual elements, 209
light theme, 185
ListBox, adding objects to Items, 208-209
PasswordBox, password entry, 251-252
popup, 316
  CoreWindowDialog, 316-319
  CoreWindowFlyout, 319-320
  MessageDialog, 321-322
  Popup, 323-325
  PopupMenu, 322-323
  ProgressRing, 326
range, 313
  ProgressBar, 314
  Slider, 314-316
RichEditBox, text editing, 248-250
TextBox, 240
    AcceptsReturn versus TextWrapping, 241
    software keyboard, 243-248
    spelling and text prediction, 241-242
text selection, 243
ToggleSwitch, 326-327
WebView, 327-330
XAML restyling
    styles, 410-418
templates, 418-428
    visual states, 428-438
Convert method (Binding), 454
Converter property (Binding), 454
converters
  SVG-to-XAML, 347
  value, 451-454
Coordinate property, 534
CoreDispatcher, 301
CoreWindowDialog popup control, 316, 318-319
CoreWindowDialog range control, 319
CoreWindowFlyout popup control, 319-320
crashes, 152
CreateAsync method (BitmapEncoder), 276
CreateForInPlacePropertyEncodingAsync method, 283
CreateForTranscodingAsync method, 280
CreateHtmlFormat, 487
CreatePrintTask, 495
CreationCollisionOption enumeration value, 465
CroppedAspectRatio property (PhotoSettings property), 296
CroppedSizeInPixels property (PhotoSettings property), 296
cropping photos, 295
cropping UI, 295
CrossSliding events, 139
CrossSliding gesture, 129
current dimensions, layout, 66
CurrentAppSimulator class, testing Windows Store features, 172-174
CurrentMarket property (ListingInformation object), 169
CurrentOrientation property, 70
CurrentStateChanged event (MediaElement), 289
custom animations, 382-383
  ColorAnimation class, 387, 391-393
data types, 382
dependency properties, 383
  DiscreteXXXKeyFrame class, 398-399
  DoubleAnimation class, 382-385
  DoubleAnimationUsingKeyFrames class, 395-397
duration, 384-385
  EasingXXXKeyFrame class, 399
  From property, 385-387
  independent versus dependent, 383-384
  ITranslateTransform, 384
  keyframes, 395
    discrete, 398-399
    easing, 399
    linear, 395-396
    spline, 397
  KeySpline class, 397
  PointAnimation class, 382, 387
  property paths, 391-393
  ScaleTransform, 384
storyboards, Timeline class properties, 393, 395
  AutoReverse, 388
  BeginTime, 388
  FillBehavior, 390
  RepeatBehavior, 389-390
  SpeedRatio, 388
  To property, 385-387
custom attributes (.NET), 34
custom commands, CoreWindowDialog, 318
custom controls, software keyboard, 248
custom search suggestions, 483-485
customizing
  background tasks
    applying conditions, 527
    event progress, 525
    registering task, 525
    triggers, 525-526
collection views, 455
groupings, 455-459
navigating, 459
data flow, 442-443
images
  stretching with nine-grid, 257, 259
playback, 288
ScrollViewer, 90-93
Slider ToolTip, 316
splash screen, 10-11
tile, 12-13
customSettingsPane control, 505-507

data
  App data, 461
    app files, 464-466
    app settings, 462-464
  network access, 469
    background download, 471
    connection information, 474
    HTTP requests, 469-471
    receiving via sockets, 471
    syndication, 471, 473
  User data, 466-467
    file picker, 467-468
    libraries and folders, 468
data binding, 439-440
  binding to collections, 444-446
C#, 441
  customizing data flow, 442-443
  data templates, 448
  items controls, 209
  Mode property (Binding object), 442
  OneWay binding, 443
  to plain properties, 442
  RelativeSource property (Binding object), 441
  sharing source with DataContext, 443-444
  source property (Binding object), 440
  target property (Binding object), 440
  TwoWay binding, 443
data flow, customizing, 442-443
data package managers, 486
data packages, 486
data templates, 447-451
data types
  Brush, 348
    ImageBrush, 355-356
    LinearGradientBrush, 349-355
    properties, 348
    SolidColorBrush, 348-349
    WebViewBrush, 358-363
custom animations, 382
Geometry, 340
  EllipseGeometry class, 340
  GeometryGroup class, 340, 344-345
  LineGeometry class, 340
  PathGeometry class, 340-341
  RectangleGeometry class, 340
  strings, 346-348
  Transform property, 341, 345
Shape, 334
  Ellipse class, 335
  Fill property, 334-335
  Line class, 336
  Path class, 338
  Polygon class, 337
  Polyline class, 336-337
  Rectangle class, 334-335
  Stroke property, 334-335
data virtualization (ListView control), 219
DataContext property, sharing source with, 443-444
DataRequested event, 486, 488
DataTemplate, properties for attaching data templates, 447-451

DateTime data type, 273

DateTimeOffset data type, 273

Debug Location toolbar, 155

declarations, extensions
   AutoPlay content, 512-514
   background tasks, 520-527
   file type associations, 516-518
   protocol, 518-519
   user account picture change, 509-511
   Windows contact picker, 514-516

Declarations tab (package manifest), 18

declaring XML elements, 28

DecodePixelHeight property (BitmapImage), 255

DecodePixelWidth property (BitmapImage), 255

decoding images
   BitmapDecoder class, 267-271
   reading metadata, 271
      BitmapProperties from a decoder, 273-275
      ImageProperties from a file, 272-273
      metadata query language, 275-276

Default parameter (ScrollIntoView), 217

DefaultPlaybackRate, 288-289

defaults, RichTextBlock, 235

dependency properties, 101-102
   animation classes, 383
   attached properties, 107-108
   change notification, 104
   implementation, 102-104
   multiple provider support, 106-107
   property value inheritance, 104-105

DependencyObject class, 103

DependencyProperty.Register method, 103

dependent animations, 366
   versus independent, 383-384

dependent packages, 256

Description property, 487

Description property (ListingInformation object), 169

design, AppBar control, 198-200

DesiredSize property, 49

devices. See also sensors
   accelerometer, 529-531
      shake detection, 532
      tossing motion, 531-532
   capabilities, 16-17
   connected standby mode, 153
   GPS, 534-535
   location, 534-535
   peer
      communication, 537-538
      finding, 537-538
   receiving messages, 536-537
   sending messages, 536-537
   source, 492
   target, 492

Devices charm, 492-493
   Play To feature, 501-502
   printing, 493-498
      adding custom options, 499-500
      changing default options, 498
      configuring displayed options, 499

Devices pane, 492-493

dictionaries, 38
   themed, XAML control restyling, 415, 417

dimensions
   accelerometer, 529-531
      shake detection, 532
      tossing motion, 531-532
   layout, 66

direct child, 187

Disabled state, ScrollBarVisibility, 91

Disabled value (NavigationCacheMode property), 177
discrete keyframes, 398-399
DiscreteXXXKeyFrame class, 398-399
Dispatcher property (CoreDispatcher), 301
dispatcherTimer class, 404
displaying text, TextBlock, 227-229
DisplayMemberPath property, 445
DisplayProperties.ResolutionScale property, 71
DisplayRequest object, 294
Document property (RichEditBox), 248
document sources, 495
DocumentProperties class, 272
Documents Library, 16
DocumentSource property, 495
Double.IsNaN method, 49
DoubleAnimation class, 382-383
Duration property, 384-385
DoubleAnimationUsingKeyFrames class, 395-397
DoubleTapped gesture event, 133
DownloadProgress event (BitmapImage), 255
DownloadProgress property, 290
DownloadProgressOffset property (MediaElement), 290
drag and drop implementation, pointers, 120, 122
DragEnter mouse-specific events, 140
draggable thumb, Slider range control, 314
Dragging event (GestureRecognizer class), 139
DragItemStarting event (ListView control), 219
DragItemThemeAnimation, 380
DragLeave mouse-specific events, 140
DragOverThemeAnimation, 38
drivers, connected standby mode, 153
Drop mouse-specific events, 140
drop-downs, 213
DropDownClosed event (ComboBox control), 213
DropDownOpened event (ComboBox control), 213
DropTargetItemThemeAnimation, 381
Duration property, animation themes, 381
duration, custom animations, 384-385
Duration property (DoubleAnimation class), 384-385
dynamic images, generating with WriteableBitmap, 260-263

E

Easeln (EasingMode), 400, 403-404
EaseOut (EasingMode), 400, 403-404
easing functions, 400, 403-404
BackEase, 403
BounceEase, 403
CircleEase, 403
ElasticEase, 403
ExponentialEase, 403
power easing, 400-401
SineEase, 403
easing keyframes, 399
EasingFunction property, 400-401
EasingMode, 400, 403-404
EasingXXXKeyFrame class, 399
EdgeGesture class, 132
EdgeUIThemeTransition, 370-371
editing text
RichEditBox control, 248-250
TextBox control, 240-242
effects
adding, 310
applying, 291
MediaElement, 291
video stabilization, 291
ElasticEase function, 403
elements, 28-29
declaring, 28
fading, 376-377
Image, 253
customizing stretching with nine-grid, 257-259
decoding images, 267-276
enabling formats of images, 253-254
encoding images, 276-280
generating dynamic images, 260-263
multiple files for multiple environments, 263-267
referencing files with URIs, 254-257
transcoding data, 280-283
naming, 41-42
eligibility for focus, UIElements receiving keyboard input, 146-147
Ellipse class, 335
EllipseGeometry class (Geometry data type), 340
Enabled value (NavigationCacheMode property), 177
EnableDependentAnimation property, 383
EncodeUserSelectedFile method, 276
encoding images (BitmapEncoder class), 276-277
writing metadata, 279-280
writing pixel data, 277-279
EndPoint property (LinearGradientBrush), 350-352
enhanced standard RGB color space (scRGB), 348
Enterprise Authentication network capability, 17
EntranceThemeTransition, 368-369
enumerations
  BindingMode, 442
  VideoEncodingQuality, 307-308
environments, automatic scaling, 263-267
escape sequences, &#xHexValue, 202
EvenOdd (FillRule property), 343-344
event attribute, 28
event handlers, SizeChanged, 66
events
  AccountPictureChanged, 509
  Activate (Windows class), 156
  Activated (Window.Current property), 153
  AddPages, 495
  bubbling, 108, 110-113
  Click, 188-189
    RepeatButton control, 191
  CurrentStateChanged (MediaElement), 289
  DataRequested, 486, 488
  DragItemStarting (ListView control), 219
  DropDownClosed (ComboBox control), 213
  DropDownOpened (ComboBox control), 213
  gestures, 133, 140
  GetPreviewPage, 495
  keyboard input, 142-143
  LayoutUpdated, 66
  LicenseChanged, 170
  manipulation, 134
  manipulations, 134-138
  MarkerReached, 290
  MediaCapture, 300-301
  MediaElement, 289-290
  ordering, 29
  pointers, 118, 120
    keyboard modifiers, 145-146
    PointerWheelChanged, 138-139
  PrintTaskRequested, 495
  ReadingChanged, 530-532
  routed, 108-110
    halting bubbling, 111-113
    leveraging event bubbling, 110-111
  SelectionChanged
    ComboBox control, 214
    Selector subclass, 208
Suspending (Application class), 152-153
deferrals, 154
handling, 153-154
Tapped, 188
TextChanged (TextBox), 440
VisibilityChanged (Windows class), 156
Execute method, 114
execution states, 150-151
Exit method (Application class), 152
Expansion property (ManipulationData event), 135
ExpirationTime property, 548
explicit runs, text content, 231
explicit sizes, 49
ExponentialEase function, 403
extended buttons, 139
extensions, 18
AutoPlay content, 512-514
AutoPlay device, 512-514
background tasks
audio, 520-523
customizing, 523-527
Binding
binding to collections, 444-446
binding to plain properties, 442
C#, 441
customizing data flow, 442-443
data templates, 448
Mode property, 442
OneWay binding, 443
RelativeSource property, 441
sharing source with DataContext, 443-444
source property, 440
target property, 440
TwoWay binding, 443
file type associations, 516-518
protocol, 518-519
user account picture change, 509-511
.vsix, 292
Windows contact picker, 514-516

F
F5, 8
FadeInThemeAnimation, 379-381
FadeOutThemeAnimation, 376-381
fading elements, 376-377
Failed even (MediaCapture), 300
file picker, User data, 467-468
file type associations, extensions, 516-518
FileOpenPicker class, 467
files
application state, 254
capabilities, 16
referencing with URIs, 254-257
saving, local file system, 464
FileSavePicker class, 467
Fill property (Shape data type), 334-335
FillBehavior property (Timeline class), 390
filled view state, 67-69
FillRule property
GeometryGroup class, 344
PathGeometry, 343-344
FindElementsInHostCoordinates method, 124
finding peer devices, 537-538
FindName method, 41
Flat line caps, 338
flight simulators, inclinometer, 532
flipped orientations, 10
FlipView control, 221-223
flow direction, 52
FlowDirection property (FrameworkElements), 54-55
FlushAsync, 277
flyouts, 213
focus eligibility, UIElements receiving keyboard input, 146-147
Focus method, 147
focus rectangle, 147
FocusManager.GetFocusedElement method, 147
FocusState property, 147
folders, User data, 468
FontFamily property (TextBlock), 227
fonts, Segoe UI Symbol, 202
FontSize property (TextBox), 227
FontStretch property (TextBox), 227
FontStyle property (TextBox), 227
FontWeight property (TextBox), 227
Format property
  PhotoSettings property, 296
  VideoSettings property, 298
FormattedPrice property (ListingInformation object), 169
formatting text, 235-237
forward navigation, 176-177
Frame
  Content property, 177
  embedding frames, 180-182
  GetNavigationState method, 179
  GoBack method, 176-177
  GoForward method, 176-177
  Navigate method, 175-177
  SetNavigationState method, 179
Frame property, 175-176
FrameCount property (BitmapDecoder), 268
Framedl property (PointerPoint class), 118
FrameworkElement class, 41
  automaton IDs, 215
FrameworkElements
  position properties, 52
    alignment, 52-53
    content alignment, 53-55
  size properties
    Height and Width properties, 48-49
    Margin and Padding properties, 50-51
free trials, Windows Store, 166-167
From property, custom animations, 385-387
FromHorizontalOffset property (EntranceThemeTransition), 368
FromVerticalOffset property (EntranceThemeTransition), 368
Fullscreen view states, 67, 69
functions, easing, 400, 403-404
  BackEase, 403
  BounceEase, 403
  CircleEase, 403
  ElasticEase, 403
  ExponentialEase, 403
  power easing, 400-401
  SineEase, 403

G
games
  controllers, handling input, 116
  pausing, 153

generating dynamic images with WriteableBitmap, 260-263
Geolocator class, 534-535
geometries, triangles, 344
Geometry data type, 340
  EllipseGeometry class, 340
  GeometryGroup class, 340, 344
  FillRule property, 344
  triangles, 345
LineGeometry class, 340
PathGeometry class, 340-341
  FillRule property, 343-344
  PathFigures, 341, 343
  PathSegments, 341-342
RectangleGeometry class, 340
strings, 346-348
Transform property, 341, 345

GeometryGroup class (Geometry data type), 340, 344
  FillRule property, 344
  strings, 346-348
  triangles, 345

Geoposition object, 534-535
gesture recognizer, 128
GestureRecognizer class, 128-132, 139
gestures, 127
  EdgeGesture class, 132
  events, 133, 140
  GestureRecognizer class, 128-132

GetCharacterRect method (TextPointer class), 235
GetChild method, 109
GetChildCount method, 109
GetCurrentPoint method, 118
GetCurrentReading method, 530
GetFoldersAsync method, StorageFolder, 468
GetForCurrentView method, 132, 138
GetFrameAsync method (BitmapDecoder), 268
GetHtmlFormatAsync, 492
GetIntermediatePoints method, 118-119
GetNavigationState method (Frame), 179
GetParent method, 109

GetPosition method, 119-120
GetPositionAtOffset method (TextPointer class), 234

GetPositionFromPoint method (RichTextBlock), 235
GetPreviewAsync method (BitmapDecoder), 271
GetPreviewPage event, 495
GetPropertiesAsync method, 274
GetStreamAsync method, 470
GetThumbnailAsync method, 271
GetThumbnailAsync method (StorageFile), 267
GetValue method, 103
GlobalOffsetX property, 64
GlobalOffsetY property, 64
GlobalOffsetZ property, 64
GoBack method (Frame), 176-177
GoForward method (Frame), 176-177
GoHome method (LayoutAwarePage class), 178
GPS device, 534-535

gradients
  colors, 391-393
  radial, 355
  transparent colors, 354

graphical badges, 550
graphics (vector), 333
  Bézier curves
    S-line shapes, 341
    U-line shapes, 341

Brush data type, 348
  ImageBrush, 355-356
  LinearGradientBrush, 349-355
  properties, 348
  SolidColorBrush, 348-349
  WebViewBrush, 358-363

Geometry data type, 340
  EllipseGeometry class, 340
  GeometryGroup class, 340, 344-345
  LineGeometry class, 340
  PathGeometry class, 340-341
RectangleGeometry class, 340
strings, 346-348
Transform property, 341, 345
scalability, 364
Shape data type, 334
   Ellipse class, 335
   Fill property, 334-335
   Line class, 336
   Path class, 338
   Polygon class, 337
   Polyline class, 336-337
   Rectangle class, 334-335
   Stroke property, 334-335
strokes, 338-340
gravitational force, accelerometer dimensions, 530
Grid App, 7
Grid panel, 75-78
   comparison to other panels, 82
      mimicking Canvas panel, 82-83
      mimicking StackPanel panel, 83
   sizing rows and columns, 79-82
Grid/, 211
GridLength structures, 82
GridLength.Auto property (Grid panel), 82
GridUnitType enumeration, 82
GridView, 445-446, 452, 454
GridView control, 219-221
groupings, collection views, 455-459
GroupName property (RadioButtons control), 193
groups
   RadioButtons control, 193
      state groups, 429
gyrometer, 532
Gyrometer class, 532

H

Handled method, 119
Handled property (KeyRoutedEventArgs instance), 142
handlers, OnSuspending, 153-154
HasOverflowContent property (RichTextBlock), 240
Header property
   GridView, 452-454
      ListView control, 217
header-embedded script commands, 290
HeaderTransitions property (ListViewBase), 367
Heading property (Coordinate property), 535
HeadingMagneticNorth property (CompassReading type), 533
HeadingTrueNorth property (CompassReading type), 533
Height property, FrameworkElements, 48-49
heuristics, 235
Hidden state, ScrollBarVisibility, 91
hiding software keyboard, 248
high-contrast themes, 187
hit testing, pointers, 123-125
Holding gesture, 128
   event, 133
homeregion-XXX resource qualifier, 265
HorizontalAlignment property (FrameworkElements), 52-53
HorizontalAlignment property (VariableSizedWrapGrid panel), 86
HorizontalContentAlignment property, 53
HorizontalScrollBarVisibility property (ScrollViewer), 91
HorizontalScrollMode property (ScrollViewer), 93
HorizontalSnapPointsAlignment property (ScrollViewer), 94
HorizontalSnapPointsType property (ScrollViewer), 93
HorizontalWheelPresent property (MouseCapabilities class), 138
HtmlFormatHelper.CreateHtmlFormat, 492
HTTP requests, network data access, 469-471
HyperlinkButton control, 189-190

IBackgroundTask, implementation, 523-524
Iconnection information, network data access, 474
icons, creating, 202
Icustomizing, background tasks
  IbackgroundTask implementation, 523-524
  registering task, 524-525
identity capabilities, 17
Idevices, AutoPlay, 512-514
Ignorable attribute (markup compatibility XML namespace), 31
IKeySpline class, 397
Image element, 253
  customizing stretching with nine-grid, 257, 259
  decoding images
    BitmapDecoder class, 267-269, 271
    reading metadata, 271-276
  enabling formats of images, 253-254
  encoding images, BitmapEncoder class, 276-280
  generating dynamic images, 260-263
  multiple files for multiple environments, automatic scaling, 263-267
  referencing files with URIs, 254-257
  transcoding data, 280-283
image files, customizing tile, 12-13
ImageBrush, 355-356
ImageFailed event (BitmapImage), 254-255
ImageOpened event (BitmapImage), 254-255
ImageProperties class, 272-273
implicit runs, text content, 231
implicit styles, XAML control restyling, 415
in-app purchases, 169
  enabling, 170-171
  ProductLicenses property, 169-170
in-memory data
  App data, 461
    app files, 464-466
    app settings, 462-464
  network access, 469
    background download, 471
    connection information, 474
    HTTP requests, 469-471
    receiving via sockets, 471
    syndication, 471, 473
  User data, 466-467
    file picker, 467-468
    libraries and folders, 468
inclinometer, 532
independent animations, 366, 383-384
index markers, 290
indexed files, search suggestions, 485-486
indexing local data, 465
inertia, manipulation events, 137-138
InitializeAsync overload, 299-300
  AudioDeviceId property, 299
  VideoDeviceId property, 299
InitializeComponent call, 19
Inline (TextElements), 232
Inlines property (TextBox), 229
InlineUIContainer element, 236-237
input, 115
  game controllers, 116
  keyboard input, 142
    eligibility for focus, 146-147
    events, 142-143
key states, 143-145
modifiers in pointer events, 145-146
mouse input, 138
  Dragging event (GestureRecognizer class), 139
  mouse-specific gesture routed events, 140
  MouseCapabilities class, 138
  MouseDevice class, 138
  PointerWheelChanged pointer event, 138-139
pen input, 140-142
touch input, 116
gestures, 127-133
  manipulations, 133-138
  pointers, 116-127
input pane, 243
input scope, 244
InputPane.GetForCurrentView method, 248
InputScope property (TextBox), 244
instances, MediaCaptureInitializationSettings, 299
InteractiveSession.IsRemote property, 291
interactivity, 101
  commands, 113-114
  dependency properties, 101-102
  attached properties, 107-108
  change notification, 104
  implementation, 102-104
  multiple provider support, 106-107
  property value inheritance, 104-105
routed events, 108-110
  halting bubbling, 111-113
  leveraging event bubbling, 110-111
Internet (Client & Server) network capability, 17
Internet (Client) network capability, 17
IRandomAccessStream, 256
IsAudioOnly property (MediaElement), 288
IsBarrelButtonPressed property
  (PointerPointProperties class), 141
IsChecked property
  CheckBox control, 192
  RadioButton control, 192
  ToggleButton control, 191
IsClosed, 342
IsDropDownOpen property (ComboBox control), 213
IsEditable property (ComboBox control), 214
IsEnabled property, 123
IsEraser property (PointerPointProperties class), 141
IsExtendedKey property, 142
IsHitTestVisible property, 123
IsHorizontalMouseWheel event, 139
IsHorizontalScrollChainingEnabled property
  (ScrollViewer), 93
IsInContact property (Pointer class), 117
IsInContact property (PointerPoint class), 118
IsIndeterminate property, 314
IsInRange property (PointerPointProperties class), 118, 141
IsInRange property (Pointer class), 117
IsInverted property (PointerPointProperties class), 141
IsKeyReleased property, 143
IsLeftButtonPressed property
  (PointerPointProperties class), 139
IsLightDismissEnabled property, 325
IsKeyDown property, 143
IsMiddleButtonPressed property
  (PointerPointProperties class), 139
IsSockets, network data access, 471
IsPointerOver property, 188
IsPressed property, 102, 188
IsReadOnly property (TextBox), 243
IsRightButtonPressed property
  (PointerPointProperties class), 139
IsScrollInertiaEnabled property (ScrollViewer), 93
IsSpellCheckEnabled (TextBox), 241
IsStaggeringEnabled property (EntranceThemeTransition), 368
IsSticky property (AppBar control), 198
IsTextPredictionEnabled (TextBox), 241
IsTextSelectionEnabled property (TextBlock), 233-234
IsThreeState property (ToggleButton control), 191
IsVerticalScrollChainingEnabled property (ScrollViewer), 93
IsXButton1Pressed property (PointerPointProperties class), 139
IsXButton2Pressed property (PointerPointProperties class), 139
Is syndication, network data access, 471, 473
IsZoomChainingEnabled property (ScrollViewer), 98
IsZoomInertiaEnabled property (ScrollViewer), 98
item containers, 209
ItemContainerTransitions property, 367
ItemHeight property (VariableSizedWrapGrid panel), 85
Itemtemplates, XAML control restyling
setting Template inside a Style, 427-428
target control properties, 422-426
Items collection, properties, 446
items controls, 207
automaton IDs, 215
ComboBox, 213
DropDownClosed event, 213
DropDownOpened event, 213
IsDropDownOpen property, 213
IsEditable property, 214
keystrokes, 214
SelectionChanged event, 214
data binding, 209
FlipView, 221-223
GridView, 219-221
ItemContainerTransitions property, 367
items panels, 210-212
ItemsControl class, 208-209
ListBox, 214-215
SelectedItems property, 215
SelectionMode property, 214-215
ListView, 216-217
AppBars, 218
data virtualization, 219
DragItemStarting event, 219
Header property, 217
reordering items, 219
ScrollIntoView, 217
SelectionMode property, 217-218
SemanticZoom, 223-226
visual elements, 209
items panels, 210-212
Items property (ItemsControl class), 208-209, 444
adding objects to, 208-209
ItemsControl class, 207
Items property, 208-209
ItemsSource property, 209
Selector subclass, 207-208
ItemsPanel property, 210
ItemsSource property (ItemsControl class), 209, 446
ItemWidth property (VariableSizedWrapGrid panel), 85
ITextCharacterFormat runtime interface, 249
ITextDocument runtime interface, 248
ITextParagraphFormat runtime interface, 249
ITextRange runtime interface, 249
ITextSelection runtime interface, 248
I visual states, XAML control restyling, 428
JavaScript runtime exceptions, 329

Key property (KeyRoutedEventArgs instance), 142
key states, 143-145
keyboard input, 142
  eligibility for focus, 146-147
  events, 142-143
  key states, 143-145
  modifiers in pointer events, 145-146
KeyDown event, 142
KeyDown event handler, 246
keyframes, 395
  discrete, 398-399
  easing, 399
  linear, 395-396
  spline, 397
KeyModifiers method, 119
KeyRoutedEventArgs instance, 142
KeySpline class, 397
KeyStatus property (KeyRoutedEventArgs instance), 142
keystrokes (ComboBox control), 214
keyUp event, 142
keywords, XAML, 44-45
killing (action), 152
Kind property, 132

landscape orientation, 10, 70
landscape-flipped orientation, 10, 70

language-XXX resource qualifier, 265
Latitude property (Coordinate property), 534
launch actions, 512
LaunchActivatedEventArgs instance, 156-157
Launcher class
  LaunchFileAsync method, 163-166
  LaunchUriAsync method, 164-166
LaunchFileAsync method (Launcher class), 163-166
launching
  apps, Blank App project, 8-24
  apps programmatically, 163
    customizing app launches, 165-166
    for files, 163-164
    for URIs, 164-165
launching (action), 155-156
  LaunchActivatedEventArgs instance, 156-157
  PreviousExecutionState value, 157-159
LaunchUriAsync method (Launcher class), 164-166
layout, 47, 65-66
  content overflow, 87
    clipping, 87-89
    scaling, 94-98
    scrolling, 89-94
  controlling position, 52
    alignment, 52-53
    content alignment, 53-55
  controlling size, 48
    Height and Width properties, 48-49
    Margin and Padding properties, 50-51
dimensions, 66
orientation, 70-71
panels, 71
  Canvas, 71-73
  Grid, 75-76, 78-83
  StackPanel, 74
  VariableSizedWrapGrid, 83-86
perspective transforms, 62, 64
transform classes, 55-56
  CompositeTransform, 60
  MatrixTransform, 61-62
  RotateTransform, 56-57
  ScaleTransform, 57-59
  SkewTransform, 59-60
  TranslateTransform, 60
  TransformGroup, 61
view states, 67-69
LayoutAwarePage class, 163, 178-179, 432
layoutdir-XXX resource qualifier, 266
LayoutUpdated event, 66
Leading parameter (ScrollIntoView), 217
libraries, User data, 468
LicenseChanged event, 170
licenses, 166
  enabling to be purchased, 168-169
lifecycle (app), 149
  execution states, 150-151
  transition actions, 150-151
    activating, 155-156, 159-160
    killing, 152
    launching, 155-159
  managing session state with SuspensionManager class, 160-163
  resuming, 154
  suspending, 152-154
  terminating, 155
light themes, 185-187
Line class, 336
linear keyframes, 395-396
LinearGradientBrush, 349-355
LineGeometry class (Geometry data type), 340
LineHeight property (TextBox), 228-229
LineSegment class (PathSegments), 341-342
ListBox control, 214-215
  adding objects to Items, 208-209
  ScrollViewer, 215
  SelectedItems property, 215
  SelectionMode property, 214-215
ListingInformation object, 169
  ProductLicenses property, 169-170
  ProductListings property, 171
lists, 37-38
ListView control, 216-217
  AppBars, 218
  data virtualization, 219
  DragItemStarting event, 219
  Header property, 217
  reordering items, 219
  ScrollIntoView, 217
  SelectionMode property, 217-218
ListViewBase (HeaderTransitions property), 367
Live Connect Developer Center, 471
live content, cameras, 299
live previews, video captures, 298-301
live tiles
  badges, 549-550
  secondary tiles, 550-552
  templates, 540-541
    square, 541, 543-544
    wide, 544, 547
  updates, 548
    local, 548
    periodic, 548-549
    push, 549
    scheduled, 548
LoadComponent method, 43
loading
  file variations
    automatically, 263-266
    manually, 266-267
  XAML at runtime, 40-41
local files, app files, 465
local notifications, toast notifications, 555
local settings, app settings, 462
local updates, live tiles, 548
local values (properties), 107
LocalOffsetX property, 64
LocalOffsetY property, 64
LocalOffsetZ property, 64
location, 534-535
lock screens, user status, 556-557
logic, App.xaml.cs, 22
logical direction (TextPointer class), 235
logo images, 12
Longitude property (Coordinate property), 534
loops (media), 288

\section*{M}

Main Page (Blank App), 19, 21
MainPage class, 42
MainPage.xaml.cs, printing, 493, 495
managed code, 262
ManipulationCompleted event, 134
ManipulationDelta event, 134
ManipulationInertiaStarting event, 134
manipulations, 133
  events, 134-135, 137
  inertia, 137-138
ManipulationStarted event, 134
ManipulationStarting event, 134
manual animations, 404-406
manually loading file variations, automatic scaling, 266-267
Margin property (FrameworkElements), 50-51
MarkerReached event, 290
markers, MediaElement, 290
Markers collection, 290
Markers property (MediaElement), 290
markup compatibility XML namespace, 31
markup extensions, 34, 36
Matrix property, 61
Matrix3DProjection, 64
MatrixTransform class, 61-62
MaxContacts property (PointerDevice class), 117
MaxDurationInSeconds property (VideoSettings property), 298
MaxHeight property, 48
MaximumRowsOrColumns property
  VariableSizedWrapGrid panel, 86
MaxResolution property (PhotoSettings property), 297
MaxResolution property (VideoSettings property), 298
MaxWidth property, 48
MaxZoomFactor property (ScrollViewer), 98
media
  audio, 285-286
    capture, 294, 304-305
    metadata, 289
    playback, 286-292, 294
  formats, changing, 308
  loops, 288
  video, 285-286
    background video, 291
    capture, 294, 297-304
    index markers, 290
    metadata, 289
    playback, 286-292, 294
media extensions, 285-286, 292
Media Foundation components, 285-286
media transport controls, 521-523

MediaCapture
- adjusting camera settings, 302-303
- AudioDeviceController property, 305
- capturing audio, 304-305
- capturing photos, 301-302
- capturing video, 303-304
- events, 301
  - Failed, 300
- methods
  - CapturePhotoToStorageFileAsync, 302
  - StartRecordToStorageFileAsync, 304
- switching away from applications, 300
- VideoDeviceControll property, 302-303

MediaCapture.InitializeAsync, 300

MediaCaptureInitializationSettings instance, 299

MediaElement, 286-288
- audio/video formats, 286
- CanPlayType method, 287
- content protection, 287
- CurrentStateChanged event, 289
- customizing playback, 288
- DownloadProgress property, 290
- DownloadProgressOffset property, 290
- effects, 291
- events, 289-290
- IsAudioOnly property, 288
- Makers property, 290
- markers, 290
- SetSource method, 287
- states, 289-290
- Uri options, 286

MediaEncodingProfile class, 304, 306, 308

MediaOpened, 290

MediaPlayer, 291-294

MediaTranscoder class, 305
- adding effects, 310
- changing media format, 308
- changing quality, 306-308
- PrepareStreamTranscodeAsync method, 306
- trimming files, 309-310

MergedDictionaries mechanism, 418

MessageDialog popup control, 321-322

messages
- receiving, 536-537
- sending, 536-537

metadata
- audio, 289
- reading (image formats), 271
  - BitmapProperties from a decoder, 273-275
  - ImageProperties from a file, 272-273
- metadata query language, 275-276
- video, 289
- writing, 279-280

metadata query language, 275-276

methods
- Accelerometer.GetDefault, 530
- ApplicationData.SetVersionAsync, 462
- CanPlayType (MediaElement), 287
- CapturePhotoToStorageFileAsync method (MediaCapture), 302
- CheckFeatureLicenses, 170
- Convert (Binding), 454
- Exit, 152
- Exit (Application class), 152
- GetCurrentReading, 530
- GetNavigationState (Frame), 179
- GetStreamAsync, 470
- GoBack (Frame), 176-177
- GoForward (Frame), 176-177
- GoHome (LayoutAwarePage class), 178
- LaunchFileAsync (Launcher class), 163-166
LaunchUriAsync (Launcher class), 164-166
Navigate (Frame), 175-177
OnNavigatingFrom (Page), 176
OnNavigatingTo (Page), 176
OnToggle method, 191
Page, 176
PrepareStreamTranscodeAsync (MediaTranscoder), 306
PrintTaskRequested, 496
ProximityDevice, 536
ReloadSimulatorAsync (CurrentAppSimulator class), 174
ReportStarted (ShareOperation property), 492
RestoreAsync (SuspensionManager class), 163
SaveAsync (SuspensionManager class), 163
SetNavigationState (Frame), 179
SetPreviewPage (PrintDocument), 497
SetSource (MediaElement), 287
StartRecordToStorageFileAsync (MediaCapture), 304
microphones, audio capture, 294
MinHeight property, 48
MinWidth property, 48
MinZoomFactor property (ScrollViewer), 98
modal dialog boxes, 316
Mode property (Binding object), 442
Model-View-ViewModel (MVVM) pattern, 113
motion, gyrometer, 532
mouse input, 138
  Dragging event (GestureRecognizer class), 139
  mouse-specific gesture routed events, 140
  MouseCapabilities class, 138
  MouseDevice class, 138
  PointerWheelChanged pointer event, 138-139
MouseCapabilities class, 138
MouseDevice class, 138
MouseMoved event, 138
mouseWheelDelta event, 139
ms-appdata scheme, 256
ms-appx URIs, 256
multiple files, multiple environments, 263-267
multiple provider support, dependency properties, 106-107
multithreading, 301
multitouch, 116
Music Library, 16
MusicProperties class, 272
MVVM (Model-View-ViewModel) pattern, 113
N
Name keyword, 41
Name property (ListingInformation object), 169
named elements, templates, 438
namespaces
  Windows.Devices.Input, 117
  Windows.Media, 285
  Windows.UI.Text, 248
  Windows.UI.Xaml.Controls, 71-86
  XML, 29-31
naming XAML elements, 41-42
Navigate method (Frame), 175-177
NavigateUri property, 189
navigation, 174
  back, 176-177
  Back button, 179
  basic, 175-176
  collection views, 459
  embedding frames, 180-182
  forward, 176-177
  LayoutAwarePage class, 178-179
  page caching, 177-178
  passing data, 175-176
NavigationCacheMode property (Page), 177-178
Near Field Communication. See NFC, 535
.NET custom attributes, 34
networks
capabilities, 17
data access, 469
background download, 471
connection information, 474
HTTP requests, 469-471
receiving via sockets, 471
syndication, 471-473
NFC, 535-536
chips, 535
tags, 536-537
nine-grid feature, stretching images, 257-259
NineGrid property (Image), 259
non-vector-based content, images, 253
customizing stretching with nine-grid, 257-259
decoding images, 267-276
enabling formats of images, 253-254
encoding images, 276-280
generating dynamic images, 260-263
multiple files for multiple environments, 263-267
referencing files with URIs, 254-257
transcoding data, 280-283
noninteractive video, 291
Nonzero (FillRule property), 343-344
not running (execution state), 150-151
activating, 155-156, 159-160
killing, 152
launching, 155-156
LaunchActivatedEventArgs instance, 156-157
PreviousExecutionState value, 157-159
terminating, 155
objects
Binding, 439-440
content controls, 187
DisplayRequest, 294
elements, children of, 28, 36
collection items, 37-38
content property, 36-37
type-converted values, 39-40
Geoposition, 534-535
Items property, 208-209
ListingInformation, 169
ProductLicenses property, 169-170
ProductListings property, 171
PrintManager, 495
PrintTaskOptions, 499
QuickLink, 492
RandomAccessStreamReference, 487
ShareOperation, 492
StorageFile, 446
ObservableCollection class, 445
on-screen keyboard, 243
OnCreateAutomationPeer method, 248
OneTime value (BindingMode enumeration), 442
OneWay value (BindingMode enumeration), 442-443
OnNavigatingFrom method (Page), 176
OnNavigatingTo method (Page), 176
OnSuspending handler, 153-154
OnToggle method, 191
OpenType properties, 235
order, property and event processing, 29
orientation
apps, 10
layout, 70-71
OrientationSensor, 534
  SimpleOrientationSensor, 533-534

**Orientation property**
  StackPanel panel, 74
  VariableSizedWrapGrid panel, 85

**Orientation property (PointerPointProperties class), 141**
OrientationSensor, 534

overflow (text), RichTextBlock, 237-240

OverflowContentTarget property (RichTextBlock), 238

overloads, InitializeAsync, 299-300

overscroll effect, 93

---

**P**

package display name (package manifest), 18

package manifest, 9
  Application UI tab, 10
    splash screen customization, 10-11
tile customization, 12-13
  Capabilities tab, 14-15
device capabilities, 16-17
  file capabilities, 16
  identity capabilities, 17
  network capabilities, 17
  Declarations tab, 18
  Packaging tab, 18-19

package name (package manifest), 18

packaged files, app files, 464

Packaging tab (package manifest), 18-19

Padding property (FrameworkElements), 50-51

Page
  auto-generated search results, 479
  methods, 176
  NavigationCacheMode property, 177-178

---

pages, 174
  attaching AppBar control to, 197-198
  Back button, 179
  back navigation, 176-177
  basic navigation, 175-176
  caching, 177-178
  embedding frames, 180-182
  forward navigation, 176-177
  LayoutAwarePage class, 178-179
  passing data, 175-176
  palm rejection feature, 141

panel class, 47

panels, 47, 71
  Canvas, 71-73
  ChildrenTransitions property, 367
  Grid, 75-78
    comparison to other panels, 82-83
    sizing rows and columns, 79-82
  StackPanel, 74
  VariableSizedWrapGrid, 83-86

PaneThemeTransition, 371-372

paragraphs, RichTextBlock, 236

parsing XAML at runtime, 40-41

passwords, PasswordBox control, 251-252

Password reveal button, 251

PasswordBox control, password entry, 251-252

PasswordChanged event, 251

PasswordVault class, 252

Path class, 338

PathFigures, 341-343
  FillRule property, 343-344
  triangles, 343

PathGeometry class (Geometry data type), 340-341
  PathFigures, 341-344
  PathSegments, 341-342
PathSegments, 341-342
  ArcSegment class, 341
  BezierSegment class, 341
  LineSegment class, 341-342
  PolyBezierSegment class, 341
  PolyLineSegment class, 341
  PolyQuadraticBezierSegment class, 341
  QuadraticBezierSegment class, 341
pausing games, 153
peer devices
  communication, 537-538
  finding, 537-538
pen input, 140-142
percentage sizing, 81
performance
  cached composition, 364
  shapes, 340
periodic updates, live tiles, 548-549
permissions, 14
perspective transforms, 62-64
PhotoOrVideo mode, 298
photos
  capturing, 294-297, 301-302
  cropping, 295
  live previews, 298-301
PhotoSettings property (CameraCaptureUI class), 296
pictures, user accounts, 509
Pictures Library, 16
  changing data context, 445
  GridView, 446
pixel data (images)
  GetPixelDataAsync method (BitmapDecoder), 268-271
  writing, 277-279
PixelDataProvider class, 268
plain properties, binding to, 442
plane projections, 62
PlaneProjection class, 62
Play To feature, 501-502
playback, 286
  customizing, 288
  MediaElement, 286-288
    audio/video formats, 286
    CanPlayType method, 287
    content protection, 287
    customizing playback, 288
    effects, 291
    events, 289-290
    IsAudioOnly property, 288
    markers, 290
    SetSource method, 287
    states, 289-290
    Uri options, 286
    MediaPlayer, 291-292, 294
    speed, 288-289
  PlaybackRate, 289
Playlist class, 293
PointAnimation class, 382, 387
Pointer class, 117
Pointer method, 119
PointerCanceled event, 119
PointerCaptureLost event, 119
PointerDevice class, 117
PointerDevice property (PointerPoint class), 118
PointerDeviceType property (Pointer class), 117
PointerDeviceType property (PointerDevice class), 117
PointerDownThemeAnimation, 380
PointerEntered event, 119
PointerExited event, 119
PointerId property (Pointer class), 117
PointerMoved event, 118
PointerPoint class

PointerPoint class, 117-118

PointerType class, 118

PointerPressed event, 118

PointerReleased event, 119

PointerRoutedEventArgs class, 111

PointerRoutedEventArgs instance, 119

Pointers, 116

capture and release, 120, 122

events, 118, 120

keyboard modifiers, 145-146

PointerWheelChanged, 138-139

hit testing, 123-125

Pointer class, 117

PointerType class, 117

PointerType class, 117-118

PointerTypeProperties class, 118

tracking multiple pointers, 125, 127

PointerTypeUpdateKind property (PointerTypeProperties class), 139

PointerTypeUpdateAnimation, 380

PointerTypeWheelChanged pointer event, 138-139

PolarEffect, 291

PolyBezierSegment class (PathSegments), 341

Polygon class, 337

Polyline class, 336-337

PolyLineSegment class (PathSegments), 341

PolyQuadraticBezierSegment class (PathSegments), 341

PopInThemeAnimation, 379

PopOutThemeAnimation, 380

Popup, ChildTransitions property, 367

popup controls, 316

CoreWindowDialog, 316, 318-319

CoreWindowFlyout, 319-320

MessageDialog, 321-322

Popup, 323-325

PopupMenu, 322-323

PopupMenu popup control, 323-325

PopupThemeTransition, 369-373

portrait orientation, 10, 70

portrait-flipped orientation, 10, 70

position

child layout properties, 52

alignment, 52-53

content alignment, 53-55

perspective transforms, 62, 64

transform classes, 55-56

CompositeTransform, 60

MatrixTransform, 61-62

RotateTransform, 56-57

ScaleTransform, 57-59

SkewTransform, 59-60

TranslateTransform, 60

TransformGroup, 61

Position property (PointerType class), 117

power easing functions, 400-401

prediction (text), TextBox control, 241-242

PrepareStreamTranscodeAsync method (MediaTranscoder), 306

pressed appearances, AppBar control, 205

Pressure property (PointerTypeProperties class), 141

PreviousExecutionState property (OnLaunched), 156

PreviousExecutionState value, 157-159

primary XML namespace, 31

print previews, 495

print tasks, 495

PrintDocument class, 493-498

printing, 493-498

adding custom options, 499-500

changing default options, 498

configuring displayed options, 499

PrintManager class, 493

PrintManager object, 495
PrintTaskOptions object, 499
PrintTaskRequested event, 495
PrintTaskRequested method, 496
Private Networks (Client & Server) network capability, 17
procedural code, mixing with XAML, 40
loading and parsing at runtime, 40-41
naming XAML elements, 41-42
Visual Studio support, 42-43
ProcessContent attribute (markup compatibility XML namespace), 31
processing rules, object element children, 40
ProcessMoveEvents method, 131
ProductLicenses property (ListingInformation object), 169-170
ProductListings property (ListingInformation object), 169, 171
products, 169
ProgressBar range control, 314
ProgressRing control, 326
projections, 62, 64
propdp snippet, 103
properties
  AccelerationX (AccelerometerReading class), 530
  AccelerationY (AccelerometerReading class), 530
  AccelerationZ (AccelerometerReading class), 530, 532
  AccelerometerReading class, 530
  Accuracy (Coordinate property), 534
  ActivationKind (OnLaunched), 157
  AddDeleteThemeTransition, 372-374
  AgeRating (ListingInformation object), 169
  AllowCropping property (PhotoSettings property), 296
  AllowTrimming (VideoSettings property), 298
  Altitude (Coordinate property), 534
  AltitudeAccuracy (Coordinate property), 535
  Arguments (OnLaunched), 156
  attached, 71-73
  attaching data templates, 447-451
  attributes, 28
  AudioCategory, 520
  AudioDeviceController (MediaCapture), 305
  AudioDeviceId (InitializeAsync overload), 299
  AutomationProperties.Name, 201
  BottomAppBar, 197-198
  Brush data type, 348
  Byte (Color Property), 348
  ChildrenTransitions, 367
  ChildTransitions, 367
  CivicAddress, 534-535
clearing local values, 107
  ClickMode, 188
  ColorInterpolationMode (LinearGradientBrush), 350
  Content (MessageDialog popup control), 322
  Content (Frame), 177
  ContentTemplate (ContentControl), 447
  ContentThemeTransition, 370
  ContentTransitions, 367
  Converter (Binding), 454
  Coordinate, 534
  CroppedAspectRatio property (PhotoSettings property), 296
  CroppedSizeInPixels property (PhotoSettings property), 296
  CurrentMarket (ListingInformation object), 169
  DataContext, 443-444
  Description (ListingInformation object), 169, 487
  Dispatcher (CoreDispatcher), 301
  DisplayMemberPath, 445
  DocumentSource, 495
  DownloadProgress, 290
  DownloadProgressOffset (MediaElement), 290
Duration
  animation themes, 381
  DoubleAnimation class, 384-385
EasingFunction, 400-401
EdgeUIThemeTransition, 370-371
elements (XAML), 31-33
EnableDependentAnimation, 383
EndPoint (LinearGradientBrush), 350-352
EntranceThemeTransition, 368-369
ExpirationTime, 548
Fill property (Shape data type), 334-335
FillRule
  GeometryGroup class, 344
  PathGeometry, 343-344
Format (VideoSettings property), 298
Format property (PhotoSettings property), 296
FormattedPrice (ListingInformation object), 169
Frame, 175-176
From, custom animations, 385-387
FromHorizontalOffset property (EntranceThemeTransition), 368
FromVerticalOffset property (EntranceThemeTransition), 368
Geoposition object, 534-535
GroupName (RadioButtons control), 193
Header (GridView), 452-454
Header (ListView control), 217
HeaderTransitions (ListViewBase), 367
Heading (Coordinate property), 535
HeadingMagneticNorth (CompassReading type), 533
HeadingTrueNorth (CompassReading type), 533
InteractiveSession.IsRemote, 291
IsAudioOnly (MediaElement), 288
IsChecked
  CheckBox control, 192
  RadioButton control, 192
  ToggleButton control, 191
IsDropDownOpen (ComboBox control), 213
IsEditable (ComboBox control), 214
IsIndeterminate, 314
IsLightDismissEnabled (Popup popup control), 325
IsPointerOver, 188
IsPressed, 188
IsStaggeringEnabled property (EntranceThemeTransition), 368
IsSticky (AppBar control), 198
IsThreeState (ToggleButton control), 191
ItemContainerTransitions, 367
Items, 444
Items (ItemsControl class), 208-209
  adding objects to, 208-209
ItemsPanel, 210
ItemsSource (ItemsControl class), 209, 446
Latitude (Coordinate property), 534
layout, child elements, 48-62, 64
Longitude (Coordinate property), 534
Markers (MediaElement), 290
MaxDurationInSeconds (VideoSettings property), 298
MaxResolution property (VideoSettings property), 298
MaxResolution property (PhotoSettings property), 297
Name (ListingInformation object), 169
NavigateUri, 189
NavigationCacheMode (Page), 177-178
ordering, 29
PaneThemeTransition, 371-372
PhotoSettings (CameraCaptureUI class), 296
plain, binding to, 442
PopupThemeTransition, 369-370
PreviousExecutionState (OnLaunched), 156
ProductLicenses (ListingInformation object), 169-170
ProductListings (ListingInformation object), 169, 171
RadiusX property (Rectangle class), 334
RadiusY property (Rectangle class), 334
ReorderThemeTransition, 375-376
RepositionThemeTransition, 374-375
RequestedTheme, 186-187
RoamingStorageQuota, 464
selectedIndex (Selector subclass), 207
SelectedItem (Selector subclass), 207
SelectedItems (ListBox control), 215
SelectedValue (Selector subclass), 207
SelectionMode
   ListBox control, 214-215
   ListView control, 217-218
SessionState (SuspensionManager class), 162
ShareOperation, 492
ShowError, 314
ShowPaused, 314
Source (CaptureElement class), 298
Speed (Coordinate property), 535
SplashScreen (OnLaunched), 156
StartPoint (LinearGradientBrush), 350-352
Stretch (CaptureElement class), 298
Stroke property (Shape data type), 334-335
StrokeDashArray (Shape class), 338-339
StrokeDashCap (Shape class), 338-339
StrokeEndLineCap (Shape class), 338-339
StrokeLineJoin (Shape class), 338
StrokeMiterLimit (Shape class), 338
StrokeStartLineCap (Shape class), 338
StrokeThickness (Shape class), 338
target control, XAML control restyling, 420-426
TargetName (Storyboard), 377-379
TileId (OnLaunched), 156
Timeline class, 381, 388
   AutoReverse, 388
   BeginTime, 388
   FillBehavior, 390
   RepeatBehavior, 389-390
   SpeedRatio, 388
   storyboards, 393-395
Timestamp
   AccelerometerReading class), 530-532
   CompassReading type, 533
   Coordinate property, 535
Title, 487
To, custom animations, 385-387
TopAppBar, 197-198
Transform (Geometry data type), 341, 345
Transitions (UIElement), 366-367
VideoDeviceControl (MediaCapture), 302-303
VideoDeviceId (InitializeAsync overload), 299
VideoSettings (CameraCaptureUI class), 298
Window.Current, Activated event, 153
Properties property (PointerPoint class), 118
property paths, animations, 391-393
property value inheritance, dependency properties, 104-105
proportional sizing, 80
Protocol declarations, extensions, 518-519
proximity
   NFC, 535-536
   chips, 535
tags, 536-537
   Share charm, 536
ProximityDevice methods, 536
push updates, live tiles, 549
QuadraticBezierSegment class (PathSegments), 341
quick links, 492
QuickLink object, 492

radial gradients, 355
RadialGradientBrush class, 355
Radio control, 192-193
RadiusX property (Rectangle class), 334
RadiusY property (Rectangle class), 334
randomAccessStreamReference objects, 487
range controls, 313
   ProgressBar, 314
   Slider, 314, 316
RangeBase base class, 313
RawPosition property (PointerPoint class), 118
read-only properties, layout process output, 49
reading metadata (image formats), 271
   BitmapProperties from a decoder, 273-275
   ImageProperties from a file, 272-273
   metadata query language, 275-276
ReadingChanged event, 530-532
RealTimePlayback, 289
receiving messages, NFC tags, 536-537
Rectangle class, 334-335
   RadiusX property, 334
   RadiusY property, 334
RectangleGeometry class (Geometry data type), 340
red squiggles, 242
referencing files with URIs, 254-257
RefreshCommand class, 114
relative URIs, 256
RelativeSource property (Binding object), 441
release pointers, 120-122
ReleasePointerCapture method, 120
ReloadSimulatorAsync method (CurrentAppSimulator class), 174
Remote Machine option (launching apps), 8
rendering, 447
   data templates, 447-451
   value converters, 451-454
RenderSize property, 49
RenderTransform, 89
RenderTransform property, 55
RenderTransformOrigin property, 55
reordering items (ListView control), 219
ReorderThemeTransition, 375-376
RepeatBehavior property (Timeline class), 389-390
RepeatButton control, 191
RepeatCount property, 142
reporting search results, 479-482
ReportStarted method (ShareOperation property), 492
RepositionThemeAnimation, 380
RepositionThemeTransition, 374-375
RequestAppPurchaseAsync, 168
RequestedTheme property, 186-187
RequestProductPurchaseAsync, 171
Required value (NavigationCacheMode property), 177
resolution, Windows Store apps, 9
resource lookup, XAML control restyling, 417-418
resource qualifiers, 264
RestoreAsync method (SuspensionManager class), 163
restyling XAML controls
   styles, 410-418
   templates, 418-428
   visual states, 428-438
resuming (action), 154
S

SaveAsync method (SuspensionManager class), 163
saving files, local file system, 464
scalability, vector graphics, 364
scale factor, 71
Scale property (ManipulationData event), 134
scale resource qualifier, 265
ScaleTransform, 89, 384
ScaleTransform class, 57-59
scaling
  content overflow, 94
    Viewbox element, 95-98
    ZoomMode property, 98
  image files, 263
    loading file variations automatically, 263-266
    loading file variations manually, 266-267
ScanCode property, 142
scheduled notifications, toast notifications, 555
scheduled updates, live tiles, 548
scRGB (enhanced standard RGB color space), 348
ScrollBar controls, 90
ScrollBarVisibility enumeration, 91
scrolling content overflow, 89-90
  customizing ScrollViewer, 90-93
  snap points, 93-94
ScrollView (ListView control), 217
ScrollView control, 89-90
  customizing, 90-93
  ListBox control, 215
  panning and zooming functionality, 137
Search charm, 477-479
  auto-generated search results Page, 479
  reporting search results, 479-482
  search suggestions, 483-485
  search suggestions from indexed files, 485-486
Search pane, 478
SearchResultsPage1.xaml, 478
secondary tiles, 539, 550-552
Segoe UI Symbol, 202
Select method (TextPointer class), 234
SelectAll method (TextPointer class), 234
SelectedIndex property (Selector subclass), 207
SelectedItem property (Selector subclass), 207
SelectedItems property (ListBox control), 215
SelectedText property (TextBlock), 234
SelectedValue property (Selector subclass), 207
selection
text
TextBlock, 233-235
TextBox control, 243
selection boxes, 213
SelectionChanged event
ComboBox control, 214
Selector subclass, 208
TextBlock, 234
SelectionEnd property (TextBlock), 234
SelectionMode property
ListView control, 216-217
AppBars, 218
data virtualization, 219
DragItemStarting event, 219
Header property, 217
reordering items, 219
ScrollIntoView, 217
SelectionMode property, 217-218
selectedIndex property, 207
SelectedItem property, 207
SelectedValue property, 207
SelectionChanged event, 208
selectors (template), 451
semantic zooming, 223-226
SemanticZoom control, 223-226
sending message, NFC tags, 536-537
sensors. See also devices
ambient light, 533
compass, 533
OrientationSensor, 534
SimpleOrientationSensor, 533-534
separate-stream script commands, 290
session state, 183
cache, 163
managing with SuspensionManager class, 160-163
PreviousExecutionState value, 159
SessionState property (SuspensionManager class), 162
SetAccountPictureAsync method, 509
setAccountPictureFromStreamAsync method, 509
SetAccountPicturesAsync method, 510
setAccountPicturesFromStreamAsync method, 510
SetLeft static method, 72
SetNavigationState method (Frame), 179
SetPixelData method (BitmapEncoder), 277-279
SetPreviewPage method (PrintDocument), 497
SetSource method
  BitmapSource, 256
  MediaElement, 287
Settings charm, 503-507
Settings pane, 503-507
SetTop static method, 72
SetValue method, 103
shake detection, 532
Shape class
  overuse of shapes, 340
    StrokeDashArray property, 338-339
    StrokeDashCap property, 338-339
    StrokeEndLineCap property, 338-339
    StrokeLineJoin property, 338
    StrokeMiterLimit property, 338
    StrokeStartLineCap property, 338-339
    StrokeThickness property, 338
Shape data type, 334
  Ellipse class, 335
  Fill property, 334-335
  Line class, 336
  Path class, 338
  Polygon class, 337
  Polyline class, 336-337
  Rectangle class, 334-335
  Stroke property, 334-335
Share charm, 486, 536
  share sources, 486-489
  share targets, 489-492
Share pane, 488, 490
ShareOperation object, 492
ShareOperation property, 492
sharing sources with DataContext, 443-444
ShowError property, 314
showing software keyboard, 248
ShowPaused property, 314
shrinking elements, 89
SimpleOrientationSensor, 533-534
simulator, 8-9
Simulator option (launching apps), 8
SineEase function, 403
size
  child layout properties, 48
    Height and Width, 48-49
    Margin and Padding, 50-51
  perspective transforms, 62-64
  rows and columns, Grid panel, 79-82
  transform classes, 55-56
    CompositeTransform, 60
    MatrixTransform, 61-62
    RotateTransform, 56-57
    ScaleTransform, 57-59
    SkewTransform, 59-60
    TranslateTransform, 60
    TransformGroup, 61
SizeChanged event handler, 66
SkewTransform class, 59-60
Slider range control, 314, 316
snap points, 93-94
snapped apps, 69, 468
snapped view state, 67-69
SnapPointsType enumeration, 93
software input panel, 243
software keyboard
  custom controls, 248
  showing/hiding, 248
  text entry, 243-248
SolidColorBrush, 348-349
source devices, 492
source property (Binding object), 440
Source property (CaptureElement class), 298
Source property (Image), 254
spaces, Geometry strings, 348
speed, playback, 288-289
Speed property (Coordinate property), 535
SpeedRatio property (Timeline class), 388
spelling, TextBox control, 241-242
splash screen, customizing, 10-11
SplashScreen property (OnLaunched), 156
spline keyframes, 397
SplineXXXKeyFrame class, 397
Split App, 7, 174
SplitCloseThemeAnimation, 381
SplitOpenThemeAnimation, 381
SplitPage projects, navigation, 175-176
SQLite website, 466
StackPanel panel, 74, 83, 210
staggering EntranceThemeTransition, 368
standard RGB color space (sRGB), 348
standardDataFormats class, 489
StandardStyles.xaml, 479
buttons, 201-204
XAML control restyling, 413-414
star sizing, 80
star syntax, 80
Start Debugging button, 8
Starting event (EdgeGesture), 132
StartPoint property (LinearGradientBrush), 350-352
StartRecordToStorageFileAsync method (MediaCapture), 304
states
  Button, 429, 432
groups, 429
  MediaElement, 289-290
static methods, 72
statuses, lock screen, 556-557
StorageFile class
  appending content with APIs, 465
data binding, 449
StorageFile object, 446
  accessing image metadata, 272-273
storyboards
  multiple storyboards, 390-391
  TargetName property, 377-379
  Timeline class properties, 393-395
Stretch alignment, 53
Stretch property (Viewbox), 95
Stretch property (CaptureElement class), 298
Stretch property (Image), 254
StretchDirection property (Viewbox), 95
stretching images, nine-grid, 257, 259
string key, App settings, 462
strings
  color, 348
    GeometryGroup class, 346-348
Stroke property (Shape data type), 334-335
StrokeDashArray property (Shape class), 338-339
StrokeDashCap property (Shape class), 338-339
StrokeEndLineCap property (Shape class), 338-339
StrokeLineJoin property (Shape class), 338
StrokeMiterLimit property (Shape class), 338 strokes, 338-339
StrokeStartLineCap property (Shape class), 338-339
StrokeThickness property (Shape class), 338
Style, setting Template inside, 427-428
styles, XAML control restyling, 410-412
  implicit styles, 415
  resource lookup, 417-418
  StandardStyles.xaml, 413-414
  TargetType base, 412-413
  theme dictionaries, 415-417
stylus input, 140-142

subclasses. See also classes
  Selector, 207-208
    ComboBox control, 213-214
    FlipView control, 221-223
    GridView control, 219-221
    ListBox control, 214-215
    ListView control, 216-219
    SelectedIndex property, 207
    SelectedItem property, 207
    SelectedValue property, 207
    SelectionChanged event, 208

subgroups (RadioButtons control), 193

suspending
  action, 152-153
    deferrals, 154
    handling Suspending event, 153-154
  execution state, 150-151
    resuming action, 154
    Suspending event, 152-154

Suspending event (Application class), 152-153
  deferrals, 154
  handling, 153-154

SuspensionManager class
  ISessionState property, 162
  managing session state, 160-163
  RestoreAsync method, 163
  SaveAsync method, 163

SVG-to-XAML converters, 347

SwapButtons property (MouseCapabilities class), 138

SwipeBackThemeAnimation, 380
SwipeHintThemeAnimation, 380
System.Uri, 255

tags, NFC, 536-537
tap and go, 535
TapCount property (TappedEventArgs instance), 129
Tapped event, 139, 188
Tapped gesture, 128, 133
target property (Binding object), 440
target devices, 492
TargetName property (Storyboard), 377-379
targetsize-XXX resource qualifier, 266
TargetType
  implicit styles, 415
  XAML control restyling, 412-413
tasks, registering, 524-525
TCP sockets, network data access, 471
Template property, inside Style, 427-428

templates
  attaching, 447-451
  live tiles, 540-541
    square, 541, 543-544
    wide, 544, 547
  named elements, 438
  selectors, 451
  toast notifications, 552-554
tweaking existing, 428
  XAML control restyling, 418-419
    control templates, 419-420
    target control properties, 420-422, 425-426
temporary files, app files, 466
terminating (action), 155
testing Windows Store features, 172, 174
text
  content
    TextBlock, 229-231
    TextElements, 232-233
display, TextBlock, 227-229
editing
  RichEditBox control, 248-250
  TextBox control, 240-242
formatting, RichTextBlock, 235-237
overflow, RichTextBlock, 237-238, 240
prediction, TextBox control, 241
selection
  TextBlock, 233-235
  TextBox control, 243
software keyboard, TextBox control, 243-248
underlining, 230
Text string property (TextBox), 241
TextAlignment property (TextBox), 228, 241
text selection,
TextBox control, 240, 439-440
  AcceptsReturn versus TextWrapping, 241
displaying text, 227-229
  SelectionChanged event, 234
software keyboard, 243-248
spelling and text prediction, 241-242
text content, 230
  explicit versus implicit runs, 231
  Inlines property, 229
  whitespace, 231
text selection, 233-235, 243
TextChanged event (TextBox), 241, 440
TextChanged event handler, 246
TextElements text content, 232-233
  Block, 232
  Inline, 232
TextPointer class, 234-235
  GetPositionAtOffset method, 234
  Select method, 234
  SelectAll method, 234
TextTrimming property (TextBox), 87, 228
TextWrapping property (TextBox), 228
  AcceptsReturn versus, 241
theme animations, 366, 376
  DragItemThemeAnimation, 380
  DragOverThemeAnimation, 380
  DropTargetItemThemeAnimation, 381
  Duration property, 381
  FadeInThemeAnimation, 379-381
  FadeOutThemeAnimation, 376-381
  multiple storyboards, 390-391
  PointerDownThemeAnimation, 380
  PointerUpThemeAnimation, 380
  PopInThemeAnimation, 379
  PopOutThemeAnimation, 380
  RepositionThemeAnimation, 380
  SplitCloseThemeAnimation, 381
  SplitOpenThemeAnimation, 381
  storyboards, 376-379, 390-391
  SwipeBackThemeAnimation, 380
  SwipeHintThemeAnimation, 380
  Timeline class properties, 381, 388
    AutoReverse, 388
    BeginTime, 388
    FillBehavior, 390
    RepeatBehavior, 389-390
    SpeedRatio, 388
theme dictionaries, XAML control restyling, 415-417
theme transitions, 366
  AddDeleteThemeTransition, 372-374
  applying to elements, 366-367
  ContentThemeTransition, 370
  EdgeUIThemeTransition, 370-371
  EntranceThemeTransition, 368-369
  PaneThemeTransition, 371-372
  PopupThemeTransition, 369-370
  ReorderThemeTransition, 375-376
themes
app, 186
colors, 190
dark, 185-187
high-contrast, 187
light, 185-187
user, 186-187
Thickness class, 50
threading, 301
threadPoolTimer class, 404
three-stop gradients, 391-393
ticks, Slider range control, 314, 316
TileBrush class, 355
TileId property (OnLaunched), 156
tiles
background color, 13
customizing, 12-13
ImageBrush, 355-356
live
badges, 549-550
secondary tiles, 550-552
templates, 540-544, 547
updates, 548-549
WebViewBrush, 358-363
Timeline class properties, 381, 388
AutoReverse, 388
BeginTime, 388
FillBehavior, 390
RepeatBehavior, 389-390
SpeedRatio, 388
storyboards, 393, 395
TimeSpan, 385
TimeSpan.Parse, 385
Timestamp property
AccelerometerReading class, 530, 532
CompassReading type, 533
Coordinate property, 535
PointerPoint class, 118
timing
AddDeleteThemeTransition, 372-374
ContentThemeTransition, 370
EdgeUIThemeTransition, 370-371
EntranceThemeTransition, 368-369
PaneThemeTransition, 371-372
PopupThemeTransition, 369-370
ReorderThemeTransition, 369-370
RepositionThemeTransition, 374-375
Title property, 487
To property, custom animations, 385-387
toast notifications
options for showing, 555
local, 555
scheduled, 555
templates, 552-554
ToggleButton control, 191
ToggleSwitch control, 326-327
tombstoning, 183
ToolTip control, 194-196
custom, 195
Slider, customizing, 316
ToolTipService class, 194, 196
ToolTipService class (ToolTip control), 194, 196
TopAppBar property, 197-198
tossing motion, 531-532
touch input, 116
gestures, 127
EdgeGesture class, 132
events, 133
GestureRecognizer class, 128-132
manipulations, 133-138
pointers, 116
  capture and release, 120-122
  events, 118-120
  hit testing, 123-125
  Pointer class, 117
  PointerDevice class, 117
  PointerPoint class, 117-118
  PointerPointProperties class, 118
  tracking multiple pointers, 125-127

touch keyboard, 243
TouchCapabilities class, 117
TouchConfidence property (PointerPointProperties class), 118
TouchPresent property (TouchCapabilities class), 117
tracking pointers, 125-127
TranslateTransform class, 60

transcoding
data (images), 280, 282-283
  MediaTranscoder class, 305
  adding effects, 310
  changing media format, 308
  changing quality, 306-308
  trimming files, 309-310

transform classes, 55-56
  CompositeTransform, 60
  MatrixTransform, 61-62
  perspective transforms, 62, 64
  RotateTransform, 56-57
  ScaleTransform, 57-59
  SkewTransform, 59-60
  TranslateTransform, 60
  TransformGroup, 61
Transform property (Geometry data type), 341, 345
TransformGroup class, 61
TransitionCollection element, setting transitions, 367

transitions
  actions, 150-151
    activating, 155-156, 159-160
    killing, 152
    launching, 155-159
    managing session state with SuspensionManager class, 160-163
    resuming, 154
    suspending, 152-154
    terminating, 155
  setting transitions, 367
  visual, 432-433, 437
Transitions property (UIElement), 366-367
TranslateTransform, 384
Translation property (ManipulationData event), 134
translucency, colors, 349
transparent colors, gradients, 354
triangles
  FillRule property, 344
  geometries, 344
  GeometryGroup class, 345
  PathFigures, 343
triggers, background tasks, 525-526
trimming
  content overflow, 87
  files, 309-310
Twist property (PointerPointProperties class), 141
TwoWay value (BindingMode enumeration), 442-443
type converters, 33
type-converted values, child object elements, 39-40
typed styles, 415
UI virtualization, items panels, 210
UI XML namespace, 30-31
UIElements
  embedding with RichTextBlock, 236-237
  mouse-specific gesture events, 140
  Transitions property, 366-367
  WebView rendering, 329
UIs, cropping, 295
underlining text, 230
unmanaged code, 262
updates, live tiles, 548
  local, 548
  periodic, 548-549
  push, 549
  scheduled, 548
Uri options, MediaElement, 286
URLs, ScriptNotify events, 330
UseDecoder method, 275
user data, 183, 466-467
  file picker, 467-468
  libraries and folders, 468
users
  contacts, 514-516
  status, lock screen, 556-557
  themes, 186-187

V
value converters, 451-454
values
  Disabled (NavigationCacheMode property), 177
  Enabled (NavigationCacheMode property), 177
VariableSizedWrapGrid panel, 83-86, 210-212
vector graphics, 333
  Bézier curves, 341
    S-line shapes, 341
    U-line shapes, 341
  Brush data type, 348
    ImageBrush, 355-356
    LinearGradientBrush, 349-355
    properties, 348
    SolidColorBrush, 348-349
    WebViewBrush, 358-363
Geometry data type, 340
  EllipseGeometry class, 340
  GeometryGroup class, 340, 344-345
  LineGeometry class, 340
  PathGeometry class, 340-341
  RectangleGeometry class, 340
  strings, 346-348
  Transform property, 341, 345
scalability, 364
Shape data type, 334
  Ellipse class, 335
  Fill property, 334-335
  Line class, 336
  Path class, 338
  Polygon class, 337
  Polyline class, 336-337
  Rectangle class, 334-335
  Stroke property, 334-335
strokes, 338-340
version (package manifest), 18
VerticalAlignment property
  (FrameworkElements), 52-53
VerticalChildrenAlignment property (VariableSizedWrapGrid panel), 86
VerticalContentAlignment property, 53
VerticalScrollBarVisibility property (ScrollViewer), 91
VerticalScrollMode property (ScrollViewer), 93
VerticalSnapPointsAlignment property (ScrollViewer), 94
VerticalSnapPointsType property (ScrollViewer), 93
VerticalWheelPresent property (MouseCapabilities class), 138

video, 285-286
  background, 291
capture, 294
    CameraCaptureUI class, 294, 297-298
    CaptureElement class, 298-304
    Webcams, 294
capturing photos, 301-302
formats, MediaElement, 286
index markers, 290
metadata, 289
noninteractive, 291
playback, 286
  MediaElement, 286-291
  MediaPlayer, 291-292, 294
showing live previews, 298-301
stabilization, 291
transcoding, MediaTranscoder class, 305-310
VideoDeviceController property (MediaCapture), 302-303
VideoDeviceId property (InitializeAsync overload), 299
VideoEncodingQuality
  enumeration, 307-308
  options, 307
VideoProperties class, 272
Videos Library, 16
VideoSettings property (CameraCaptureUI class), 298
view models, 449
view states, layout, 67, 69
Viewbox element, scaling content overflow, 95-98
views, collections, 455
  groupings, 455-456, 458-459
  navigating, 459
viral compatibility, 123
virtualization
  ListView control, 219
  UI, 210
virtualization support (WrapGrid panel), 86
VirtualizingStackPanel, 210
visibility properties (ScrollViewer), 92
Visibility property, 123
VisibilityChanged event (Windows class), 156
Visible state (ScrollBarVisibility), 91
visual elements, items controls, 209
visual results
  AddDeleteThemeTransition, 372-374
  ContentThemeTransition, 370
  EdgeUIThemeTransition, 370-371
  EntranceThemeTransition, 368-369
  PaneThemeTransition, 371-372
  PopupThemeTransition, 369-370
  ReorderThemeTransition, 375-376
  RepositionThemeTransition, 374-375
Visual State Manager. See VSM
visual states, XAML control restyling, 429
  responding to changes, 429, 432
  transitions, 432-433, 437-438
Visual Studio
  Debug toolbar, 155
  support for XAML and code-behind, 42-43
  Windows Store apps, 7-24
visual transitions, 432
VisualState class, 429
VisualStateManager, 428-429
  responding to visual state changes, 429, 432
  visual transitions, 432-433, 437-438
VisualTransitions, 433
VisualTreeHelper class, 109
VisualTreeHelper.FindElementInHostCoordinates method, 124
vsix extension, 292
VSM, 428-429
  responding to visual state changes, 429, 432
  visual transitions, 432-433, 437-438

W
WasKeyDown property, 143
Web images, 254
WebAuthenticationBroker class, 252
Webcams, video capture, 294
WebView control, 327-330
WebViewBrush, 358-363
white-on-transparent images, 12
whitespace
  roaming, 466
  text content, 231
WIC (Windows Imaging Component), metadata query language, 275
wide tiles, templates, 544, 547
Width property (FrameworkElements), 48-49
Window.Current property (Activated event), 153
Window.Current.SizeChanged event, 66
Windows class
  Activate event, 156
  VisibilityChanged event, 156
Windows contact picker, extensions, 514-516
Windows Dev Center dashboard
  free trials, 166-167
  in-app purchases, 169
    enabling, 170-171
    ProductLicenses property, 169-170
Windows Imaging Component (WIC), metadata query language, 275
Windows kernel, suspended state, 150
Windows Media, 285-286
Windows Presentation Foundation (WPF), 27
Windows Store, 149
  apps, 7-9
    application definition, 21-24
    Main Page, 19-21
    models, 166
      package manifest, 9-19
      business models, 166
    enabling purchase of full licenses, 168-169
    free trials, 166-167
    in-app purchases, 169
      enabling, 170-171
      ProductLicenses property, 169-170
    licenses, 166
    testing features, 172-174
Windows.ApplicationModel.Store.CurrentApp class, 166
Windows.Devices.Input namespace, 117
Windows.Foundation.Uri, 255
Windows.Media namespace, 285
Windows.Storage.ApplicationData class, 461
Windows.System.Launcher class, 163
customizing app launches, 165-166
launching apps for files, 163-164
launching apps for URIs, 164-165
Windows.System.UserProfile.UserInformation class, 509-511
Windows.UI.Text namespace, 248
Windows.UI.Xaml.Controls namespace, panels, 71
Canvas, 71-73
Grid, 75-83
StackPanel, 74
VariableSizedWrapGrid, 83-86
Windows.UI.Xaml.Documents.Typography class, 235
Windows.UI.Xaml.Media namespace
CompositeTransform, 60
MatrixTransform, 61-62
RotateTransform, 56-57
ScaleTransform, 57-59
SkewTransform, 59-60
TranslateTransform, 60
TransformGroup, 61
WPF (Windows Presentation Foundation), 27
WrapGrid, 210-212
WrapGrid panel, virtualization support, 86
WriteableBitmap, generating dynamic images, 260-263
writing
metadata, 279-280
pixel data, 277-279
X
X axis, accelerometer, 529-531
X double property, 60
XAML
apps, layout, 47-64
children of object elements, 36
collection items, 37-38
content property, 36-37
type-converted values, 39-40
control restyling
styles, 410-418
templates, 418-428
visual states, 428-438
defined, 27
elements and attributes, 28-29
keywords, 44-45
language namespace, 30-31
markup extensions, 34-36
mixing with procedural code, 40
loading and parsing at runtime, 40-41
naming XAML elements, 41-42
Visual Studio support, 42-43
motivation for use, 28
property elements, 31-33
type converters, 33
XamlTune project, 347
XML
namespaces, 29-31
templates, live tiles, 540-541
square, 541, 543-544
wide, 544, 547
toast notifications, 552-554
xml:lang attribute, 44
xml:space attribute, 44
XTilt property (PointerPointProperties class), 141
Y axis accelerometer, 529-531
Y double property, 60
YTilt property (PointerPointProperties class), 141

Z axis accelerometer, 529-531
Z order, 73
ZIndex attached property, 73
ZoomMode property (ScrollViewer), 98
zoomSnapPointsType property (ScrollViewer), 98