

THE EXPERT'S VOICE® IN WINDOWS 8

Beginning Windows Store Application Development

HTML and JavaScript Edition

*LEVERAGE YOUR EXISTING SKILLS
TO BUILD NATIVE APPLICATIONS
FOR WINDOWS 8*

Scott Isaacs and Kyle Burns

Apress®

For your convenience Apress has placed some of the front matter material after the index. Please use the Bookmarks and Contents at a Glance links to access them.



Contents at a Glance

About the Authors.....	xix
About the Technical Reviewer	xxi
Acknowledgments	xxiii
Introduction	xxv
■ Chapter 1: Welcome to a Touch-First World	1
■ Chapter 2: The Microsoft Design Language.....	13
■ Chapter 3: Designing Windows Store Applications	29
■ Chapter 4: Visual Studio 2012 and Windows Store Application Types	37
■ Chapter 5: HTML Controls	49
■ Chapter 6: WinJS Controls.....	71
■ Chapter 7: WinJS Collection Controls	93
■ Chapter 8: WinJS Custom Controls.....	113
■ Chapter 9: Building the User Interface	137
■ Chapter 10: Transitions and Animations	165
■ Chapter 11: Data-Binding Concepts	181
■ Chapter 12: Promises	223
■ Chapter 13: Web Workers	269
■ Chapter 14: Data Source Options.....	287
■ Chapter 15: Session State and Settings	317
■ Chapter 16: Files	337

■ **Chapter 17: Handling State Changes361**

■ **Chapter 18: External Libraries.....383**

■ **Chapter 19: Search and Share Contracts397**

■ **Chapter 20: Printing433**

■ **Chapter 21: Notifications and Tiles459**

■ **Chapter 22: Camera and Location485**

■ **Chapter 23: Sharing Apps in the Windows Store.....505**

Index.....517

Introduction

When I was asked a few months ago about writing a book on building applications for Windows 8 with JavaScript, my first thought was, “What can I bring to the table by writing this book?” Other authors have covered this topic, but as I reviewed a number of other books, I realized that there was a hole that I could fill by taking on this project.

I wanted to see a book that not only covered the basic technical concepts but one that would also walk a beginner through the process of building a commercial-quality, real-world application from start to finish. I wanted a book that not only presented snippets of code but one that also offered tips to improve the user experience of the reader’s applications. I wanted a book that was more of a tutorial than a reference, introducing concepts that might be new to the reader, more than digging deeper into familiar topics you might find in another book. I wanted a book that you could pick up one day and be building real applications a few days later.

Beginning Windows Store Application Development—HTML and JavaScript Edition is my attempt at providing a book that I would want to read. I’ve learned a lot while writing this book, and I hope you learn a lot by reading it.

Who This Book Is For

This book is intended for developers who have experience building web applications with HTML, CSS, and JavaScript and are interested in using their existing skills to build applications for Windows 8. It is also a good beginner’s guide for those with experience building applications for earlier versions of Windows using other technologies, such as .NET. Some coverage of HTML, CSS, and JavaScript topics is included for those with less experience, but the focus of the book is not on those technologies themselves but, instead, using those technologies to build Windows Store applications.

Throughout this book, I’ll remind you that, just like building a web application, you have the freedom to follow whatever HTML, CSS, and JavaScript practices you prefer. For example, when creating page controls, which will be first illustrated in Chapter 5 but used heavily throughout the book, I’ve opted to keep the CSS and JavaScript file created by Visual Studio in the same directory as the HTML file. Because much of my background is as a .NET developer, this is familiar, as, by default, Visual Studio keeps ASPX and ASCX files in the same directory as the C# or VB code behind files. However, you could choose to move all JavaScript files to a js folder and all CSS files to a css folder. Additionally, you might notice that the JavaScript code that I’ve written isn’t idiomatic JavaScript. Feel free to modify the code samples to match your coding style.

How This Book Is Structured

While you could certainly skip directly to the topics that interest you, this book was written to be read from beginning to end, with each chapter building upon the previous. In Chapters 1 through 3, you’ll find an overview of Windows 8 concepts, such as the touch interface and gestures, and the Microsoft design language. In Chapters 4 through 8, I’ll introduce you to working with Visual Studio, covering the various project templates available, as well as the different controls you might use in the user interface (UI) of your application.

Between Chapters 9 and 22, you'll build a fully functional, real-world application. Each chapter will cover a core concept of Windows Store application development, implemented with HTML and JavaScript. The topic of each chapter is not necessarily a prerequisite for the following chapter, but the sample application has been carefully designed such that the code samples in each of these chapters build upon those in previous chapters.

Finally, in Chapter 23, I'll cover some steps you should take to brand your application and publish it in the Windows Store.

Downloading the Code

The code for the examples shown in this book is available on the Apress web site, www.apress.com. A link can be found on the book's information page under the Source Code/Downloads tab. This tab is located under the Related Titles section of the page.

Additionally, the source code for the sample application built in Chapters 9 through 23 is available on GitHub, at www.github.com/daughtkom/Clok.

Contacting the Author

Should you have any questions or comments—or even spot a mistake you think I should know about—you can contact me through my personal web site, at www.scottisaacs.com, or on Twitter, at www.twitter.com/daughtkom.

CHAPTER 1



Welcome to a Touch-First World

In April 2010, I first heard the phrase that defined Microsoft's new strategy: "three screens and the cloud." This referred to a targeted approach to ensure that Microsoft's products were ubiquitous on mobile phones, desktop computers, and television screens and that these platforms provided a seamless experience by being held together with data in the cloud. The products represented on the three screens were Windows Phone 7, Windows 7, and Xbox 360. Microsoft still dominates the television screen, with its Xbox line accounting for approximately half of all game consoles sold worldwide and a continued focus to move that platform beyond gaming, but to me, Windows 8 brings a different meaning to three screens and the cloud—one where the three screens include phones, tablets, and PCs, all running on the Windows 8 core and tied with cloud services, as shown in Figure 1-1.



Figure 1-1. Windows 8 vision of three screens and the cloud

This book is about developing applications in this new environment, but before you start any development, you must understand the environment and how it will be used. In this chapter, I will provide some background on the user interface of Windows 8 and how users will interact with applications running on this platform. I will focus primarily on touch, but because Windows 8 is a touch-first environment and not a touch-only environment, I will also discuss when touch is not appropriate and cover alternative input methods.

Moving to More Natural Interaction

In 1985, users interacted with PCs primarily by using a keyboard, but the first Macintosh was increasing the popularity of the mouse, and Microsoft introduced Windows 1.0, which was essentially a shell that allowed people to point and click to open programs and documents instead of requiring them to remember appropriate commands to type. These mouse-based environments were successful in both the business and consumer markets and made computing accessible to the masses: by the time Windows 95 was released, PCs were not uncommon in people's homes.

Over the years, computer and software makers have flirted with the idea of a computer that could be carried anywhere in a pocket or attached to a belt. Apple attempted to realize this vision as early as 1992, but it wasn't until the mid-2000s that technology really caught up and hardware manufacturers could create small, lightweight computing devices capable of running software comparable to what would be found on the desktop. By the time hardware was ready for prime-time mobile computing by consumers, the Windows brand was firmly entrenched in the market, and Microsoft made several attempts with Windows CE, Pocket PC, and various flavors of Windows Mobile to create a mobile experience that was simply a scaled-down version of Windows. This approach yielded screens that required a lot of precision to interact with, and computers running the mobile version of Windows were largely looked at as specialized devices and not accepted by the average consumer.

The introduction of Windows Phone 7 in 2010, likely driven by the successes of Apple's iPhone three years before and the subsequent popularity of Android, discarded the notion of a tiny version of Windows and went with an entirely new user-interface concept dubbed Microsoft design language. The Microsoft design language is based on a set of core design principles focused around the user, and the finger became the primary tool for interacting with the computer. Unlike with previous versions of Microsoft's mobile operating systems, Windows Phone devices no longer shipped with the stylus being a standard component.

■ **Note** You may be familiar with the term *Metro*. Metro is a code name that Microsoft and others have previously used in a few different contexts. The Microsoft design language has been extensively referred to as the Metro design language. Additionally, the Windows Start screen has often been called the Metro interface. Applications formerly called Metro apps are officially called Windows Store applications.

With Windows 8, Microsoft has taken the opportunity to hit the “reset” button on user-interface expectations and reversed its previous strategy by bringing the interactions that are natural by necessity in the mobile world to the desktop environment, instead of taking desktop concepts to the mobile world.

Windows 8 Touch Language

With the full incorporation of touch as a first-class citizen in Windows 8, it is important to understand the language of touch gestures recognized by the operating system. This is important not only as a user of Windows 8 but even more so as a developer who wants to make sure users can learn applications as quickly as possible and have a consistent experience. The Windows touch language consists primarily of eight gestures, which I will discuss in this section.

Press and Hold

The *press-and-hold* gesture, illustrated in Figure 1-2, is analogous to the right-click gesture with a mouse. The gesture is intended to allow the user to learn something about the target or be presented with additional options, such as a context menu. This gesture is accomplished by touching a single finger to the screen and pausing until the system acknowledges the hold, often by outlining the user-interface element held.



Figure 1-2. *Press and hold*

Tap

While the press-and-hold gesture can easily be equated to a single mouse gesture, the same cannot be said for the *tap* gesture. The tap gesture, illustrated in Figure 1-3, is intended to invoke the primary action on a user-interface element. Often, this will be an action such as activating a button or following a link. The mouse gesture most closely resembling the tap gesture is the left-click, but the left-click is also used for other tasks that have their own gestures in the touch language, such as selection. This gesture is accomplished by placing a finger on the user-interface element and then immediately lifting the finger straight up.

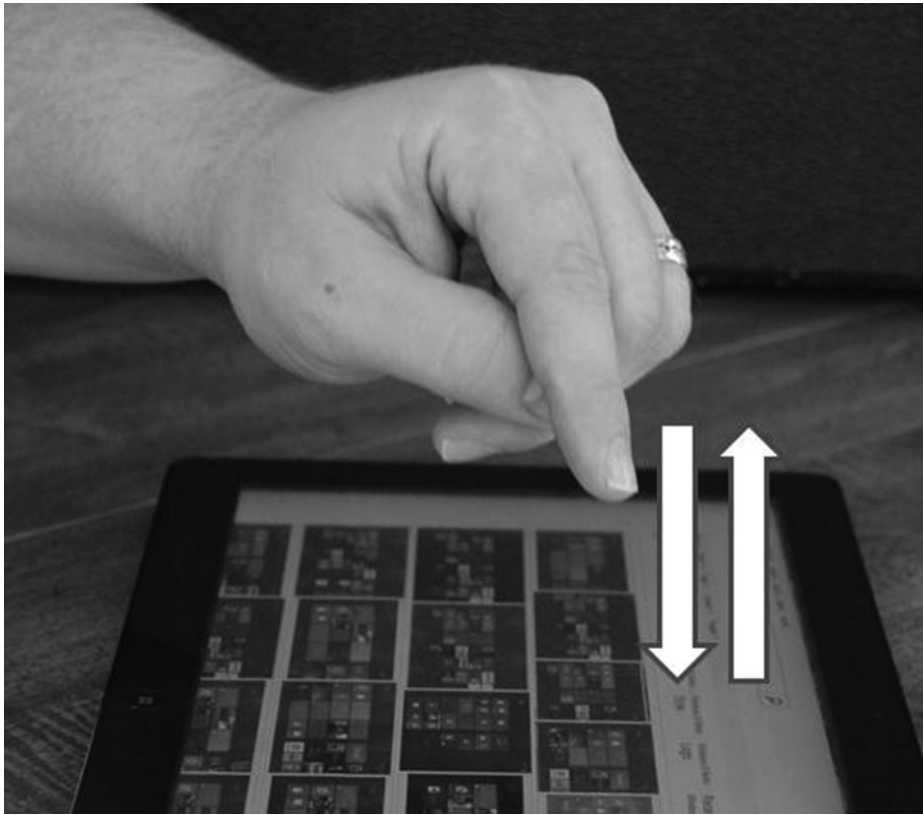


Figure 1-3. *Tap*

Slide

The *slide* gesture in the Windows touch language, shown in Figure 1-4, is used for panning or scrolling content that extends beyond the bounds of the screen or a screen section. In a mouse-driven environment, this is accomplished using scrollbars, but with touch, the slide gesture is more natural, and the scrollbar would either have to grow to the point of taking up too much real estate on the screen or be a difficult touch target. To accomplish the slide gesture, a finger is placed on the screen and then pulled up and down or side to side, depending on the orientation of the content.

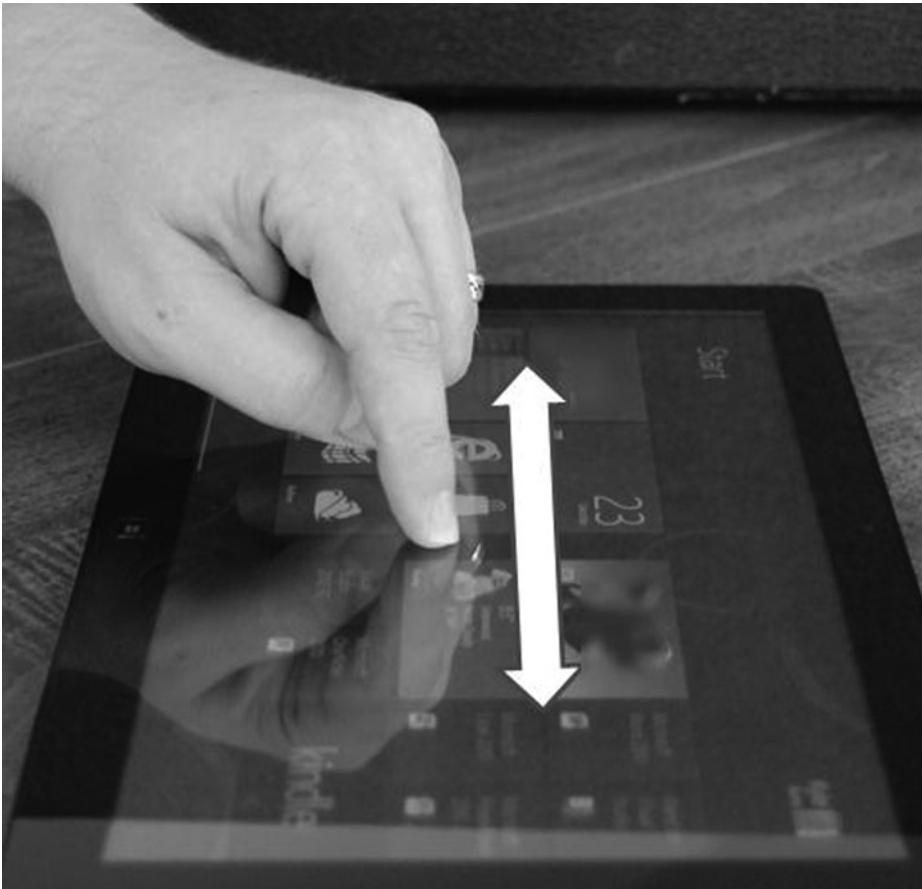


Figure 1-4. *Slide*

Swipe

The *swipe* gesture is used to communicate selection, much as left-click, Ctrl+left-click, and Shift+left-click are used when interacting with the computer using a mouse and keyboard. To achieve this gesture, shown in Figure 1-5, the finger is placed on the screen either on top of or adjacent to the item selected and then drawn through the item. The direction of the gesture depends on the orientation of the content, with horizontally oriented content being swiped vertically and vertically oriented content being swiped horizontally. The gesture going against what would be used to slide sometimes causes it to be referred to as a *cross swipe*. Use of this gesture, as opposed to a tap, eliminates the confusion that could arise when trying to accomplish multiselect scenarios with no keyboard modifier keys, such as Ctrl and Shift, that aid in mouse selection.

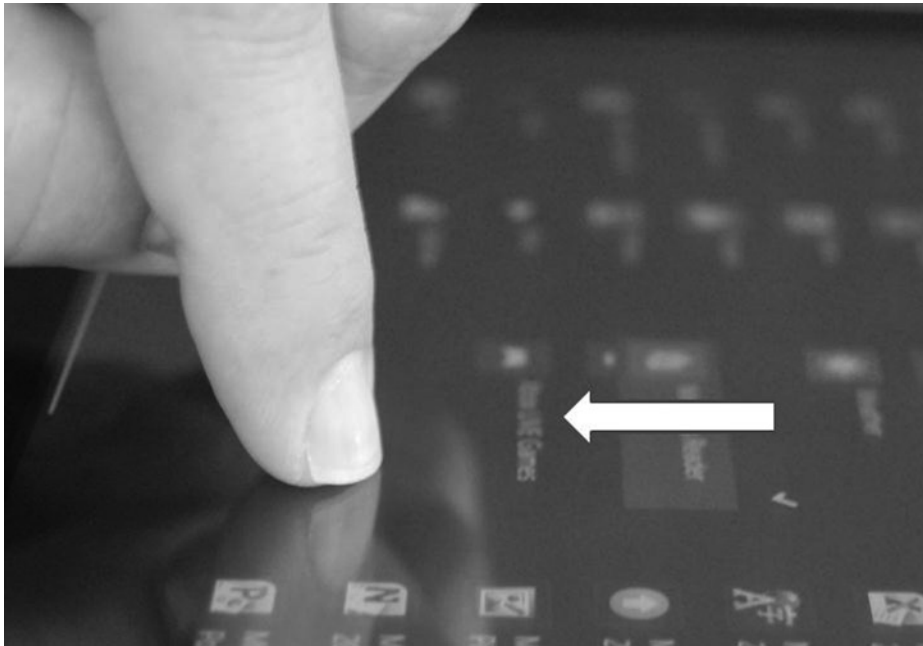


Figure 1-5. *Swipe*

Pinch

The *pinch* gesture, illustrated in Figure 1-6, does not have a direct equivalent in most mice and is considered a “zoom” gesture. The *pinch* zooms out from a narrow view with a high level of detail to a broader view with less detail. You will see in later chapters that in addition to the optical zoom, applications can take advantage of this gesture at a semantical level as well and use it to navigate summary and detail data. To accomplish the pinch gesture, two fingers are placed separated and roughly equidistant from the center of the element that is the target of the gesture, and then the fingers are slid together until either the desired zoom is met or the fingers meet.

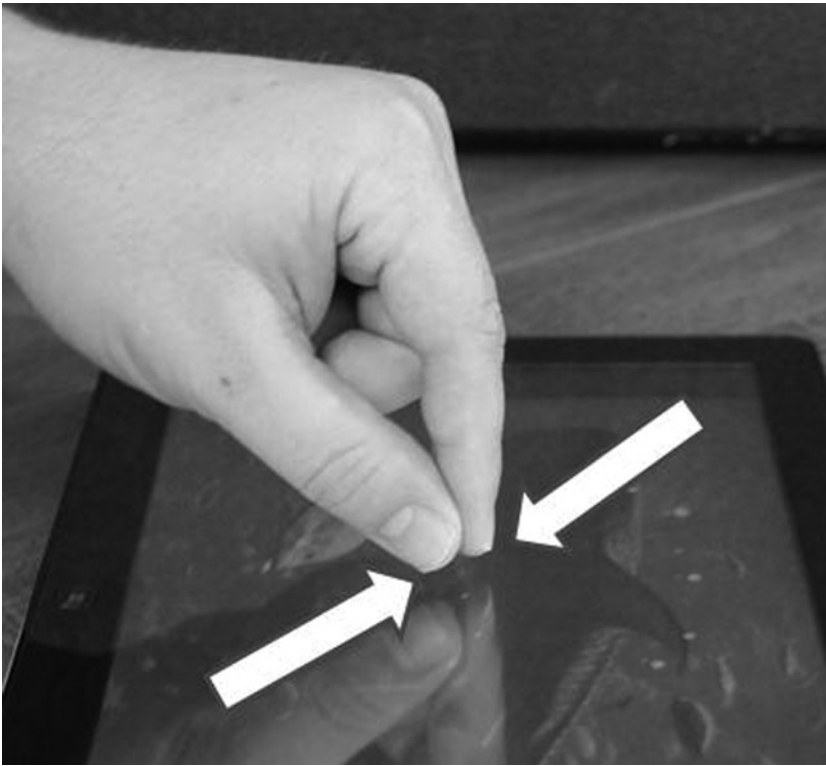


Figure 1-6. *Pinch*

■ **Note** In many cases on mice with scroll wheels, pressing the Ctrl key while scrolling performs the same action as the pinch or stretch gestures.

Stretch

The *stretch* gesture, shown in Figure 1-7, is the opposite of the pinch gesture, both in its execution and in the results. The stretch gesture is used to zoom in from a broader, less-detailed view to a narrower view with more detail. As with pinch, you will find that applications can be designed to allow the gesture to be either an optical zoom or a semantical one. To accomplish the gesture, fingers are placed together, centered on the element to be zoomed, and then are moved in opposite directions along the screen until either the desired zoom level is achieved or one of the fingers reaches the edge of the screen.

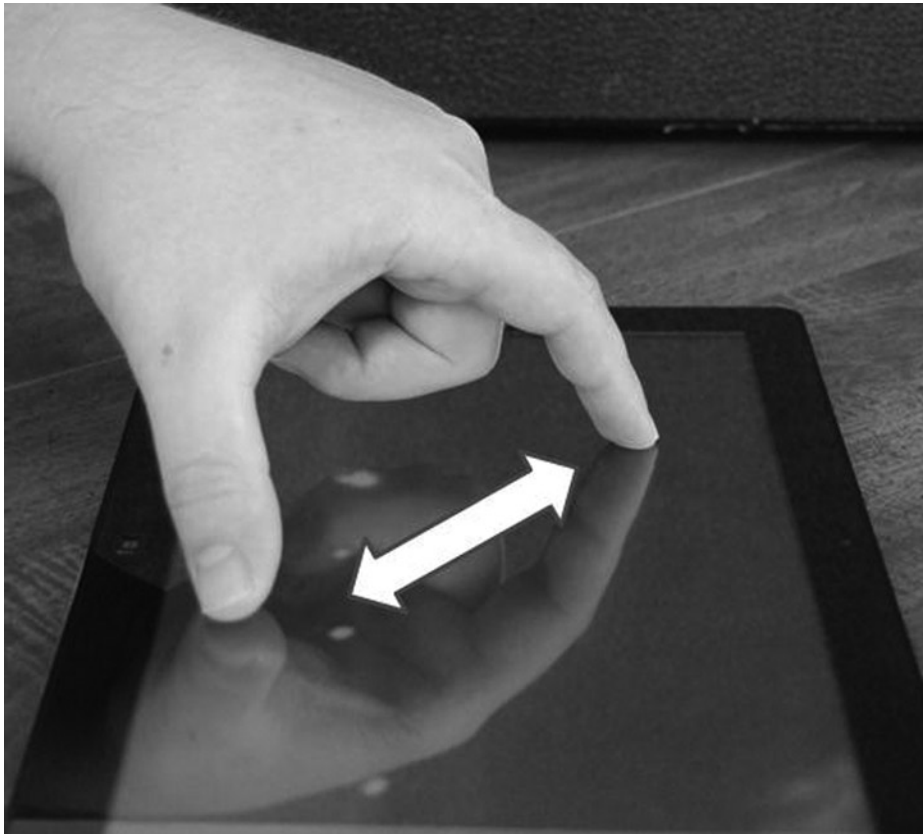


Figure 1-7. *Stretch*

Swipe from Edge

As you learn more about Windows 8 and the Microsoft design language, you will find that content is king and anything that distracts from the content is to be left off the screen. You will also find that users must be able to perform actions with the least effort possible. Windows Store applications balance these needs by placing less frequently accessed commands off the edge of the screen in what are called *app bars* and *charm bars*. The *swipe-from-edge* gesture, illustrated in Figure 1-8, is used to access these commands. To achieve the gesture, a finger is placed beyond the edge of the screen and then pulled onto the screen.



Figure 1-8. *Swipe from edge*

Turn

The *turn* gesture, illustrated in Figure 1-9, is used for rotating either the view or the content within the view. One example of where this type of gesture could be used would be in a touch version of the classic videogame Tetris, where falling blocks can be rotated to fit together. To accomplish this gesture, two fingers are placed on the screen, then either both fingers are pulled around the circumference of a circle or one is rotated around the other, which remains stationary.



Figure 1-9. *Turn*

Keys to a Successful Touch Interface

Building a successful touch interface requires careful thought and consideration on the part of the designer and developer. Many of these considerations are embedded in the design principles governing the Microsoft design language, which I will discuss in Chapter 2, but in this section, I will discuss a few concepts that are critical to touch interfaces, whether or not they use these principles.

Responsiveness

Although responsiveness is important to any application, it is especially important for users of a touch application to never be left looking at an unresponsive screen. Users are aware, even if only at a subconscious level, that a mouse pointer is a much more precise tool than the end of a finger, so if it is not readily apparent that the user's last command was accepted and is being carried out, the user is likely to feel like he or she did not hit the target and issue the command again. Responsiveness can be achieved with actions such as giving a visual clue that a long-running process has begun or ensuring that content follows the user's finger as it is dragged across the screen.

Touch Targets

As mentioned in the previous section, the mouse pointer is a far more precise tool than the human fingertip. While nothing can eliminate the possibility of the user missing targets within certain applications, using large touch targets spaced well apart is an important way to minimize missed targets. When at all possible, targets should be no smaller than 7mm square, with at least 2mm between them. As a general rule, when hitting the wrong target has severe consequences or is hard to correct, the target should be larger in proportion and should also have more space between it and other targets.

Intuitive Interface

To the end user, the best applications “just work.” Usually, this is because the application makes it easy for the user to do what needs to be done, rather than figure out how to do what needs to be done. Many desktop applications today make up for a lack of intuitiveness by providing detailed instructions in tooltips that appear as the user explores the application with a mouse pointer. Touch interfaces can still use tooltips, and the touch language defines the press-and-hold gesture for this type of learning, but it takes more effort than with a mouse, so more effort should be put into a design that clearly communicates what the user should do.

■ **Note** For additional guidance on implementing a high-quality touch-based user experience, please refer to these two MSDN articles: <http://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh465415.aspx> and <http://msdn.microsoft.com/en-us/library/windows/desktop/cc872774.aspx>.

Beyond Touch

As Windows 8 does, this chapter has placed a lot of importance on the user interacting with the computer through the use of touch gestures. It should be noted, however, that the Windows 8 user interface is referred to as *touch-first* and not as *touch-only*. Windows 8 boasts the ability to run on much of the hardware that ran on Windows XP and Windows 7 and, in many cases, will perform better because of optimizations that have been made to accommodate mobile devices. This means that even though vendors are rushing to market with innovative touch hardware, for the foreseeable future, application developers must acknowledge that many of their users will approach the application equipped only with a keyboard and mouse.

In addition to the volume of older hardware that will remain in use, it’s also important to understand that some usage scenarios simply do not translate as well to a touch environment. Users sitting for hours doing data entry are going to be much more comfortable and suffer less fatigue and injury using a keyboard and mouse than users performing the same tasks with their arm outstretched to reach a touch-screen monitor set up like most monitors today. Hardware vendors will meet this new need by continuing to innovate, and you will likely see changes such as multitouch trackpads replacing the traditional mouse and monitors that adjust to lie flat or at least angled on the desk. Additionally, expect to see devices similar to Microsoft’s Kinect device evolve and be used in even more innovative ways than seen today.

Conclusion

In this chapter, you looked at Windows 8 as the touch-first world in which your applications will live. You learned about the basic gestures that have been defined in the Windows touch language and how end users will expect applications to react to them. You also learned that regardless of what the computer of tomorrow looks like, the computer of today often looks remarkably like computers sold the day before or even five years before Windows 8 released to market and that your applications must take the users of today’s computers into account. Regardless of whether the user is interacting with hands or a mouse, Windows Store applications should be fluid, intuitive, and responsive.



The Microsoft Design Language

Beyond the basic touch principles discussed in the previous chapter, the design teams at Microsoft developed the Microsoft design language, previously referred to as Metro, which is used to guide the user-interface development for Windows Phone 7, Windows Phone 7.5, and now for Windows 8 and Windows Phone 8. The Microsoft design language was inspired by the simple, easily understood language seen in street signs in metropolitan areas and in mass transit and strives to bring this simplicity and intuitive flavor to computing. In this chapter, I will cover the elements of the Microsoft design language, show examples, and explain how Windows 8 incorporates them. Before jumping into the Microsoft design language itself, I will cover the Swiss design style, whose influence can be clearly seen in elements of the Microsoft design language.

Swiss Design Style

The Microsoft design language is influenced most by a design style known as the Swiss design style, or international typographic style, which was developed in Switzerland in the 1950s and really started coming into its own in the 1960s and 1970s.

Influence of Bauhaus

The Swiss design style was heavily influenced by the Bauhaus movement, which Walter Gropius founded in 1919 with the establishment of the art school Staatliches Bauhaus in Weimar, Germany. The guiding principle of the Bauhaus movement was that of function over form, thus favoring concise communication and stark contrast over abstract ideas and gradient transition. It promoted art and architecture designed for an industrialized society and for which it could be mass-produced. The Bauhaus movement had a significant influence on the development of modern design and architecture. Today, the web site <http://Bauhaus-online.de> is maintained by the Bauhaus Archive Berlin/Museum for Design, the Weimar Classic Foundation, and the Bauhaus Dessau Foundation (see Figure 2-1) as an effort to preserve and disseminate information about the school and educate people about the impact of the institution.



Figure 2-1. *Bauhaus building in Dessau, Germany*

Elements of Swiss Design Style

The Swiss design style is characterized by a number of elements, which I will discuss in this chapter. These elements include typography, photography, iconography, generous use of whitespace, and strict organization. Brought together, the elements produce the distinct look and feel of a work designed in the Swiss style.

Typography

Front and center among the arts influenced by Swiss design style principles is typography. The developers of the Swiss style, and those who design with it today, hold steadily that text should be clear and simple and that unnecessary adornment not only occludes the message being conveyed in the text but also actively distracts from the message. In keeping with the idea that text should be clear, concise, and simple, Swiss designs will typically feature sans-serif fonts with text left-justified and jagged (ragged) on the right. Figures 2-2 and 2-3 are examples of a newsletter designed with justified columns and a serif font (Times New Roman) followed by the same newsletter designed using a sans-serif font (Helvetica) and left-justified to align with Swiss design style principles. Note the marked difference, specifically in the typeface, between the two examples and how the sans-serif typeface produces a cleaner look. The headlines are especially good examples of this.

LEAD STORY HEADLINE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc.

Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor. Maecenas mattis. Sed convallis tristique sem. Proin ut ligula

vel nunc egestas porttitor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. Fusce ac turpis quis ligula lacinia aliquet. Mauris ipsum. Nulla metus metus, ullamcorper vel, tincidunt sed, euismod in, nibh. Quisque volutpat condimentum velit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nam nec ante.

Sed lacinia, urna non tincidunt mattis, tortor neque adipiscing diam, a cursus ipsum ante quis turpis. Nulla facilisi. Ut fringilla. Suspendisse potenti. Nunc feugiat mi a tellus consequat imperdiet. Vestibulum sapi-

en. Proin quam. Etiam ultrices. Suspendisse in justo eu magna luctus suscipit. Sed lectus. Integer euismod lacus luctus magna. Quisque cursus, metus vitae pharetra auctor, sem massa mattis sem, at interdum magna augue eget diam. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Morbi lacinia molestie dui. Praesent blandit dolor.

Sed non quam. In vel mi sit amet augue congue elementum. Morbi in ipsum sit amet pede facilisis laoreet. Donec lacus nunc, viverra nec, blandit vel, egestas et, augue. Vestibulum tincidunt malesuada tellus. Ut ultrices

Figure 2-2. Mock newsletter in non-Swiss style

LEAD STORY HEADLINE

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc.

Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem

at dolor. Maecenas mattis. Sed convallis tristique sem. Proin ut ligula vel nunc egestas porttitor. Morbi lectus risus, iaculis vel, suscipit quis, luctus non, massa. Fusce ac turpis quis ligula lacinia aliquet. Mauris ipsum. Nulla metus metus, ullamcorper vel, tincidunt sed, euismod in, nibh. Quisque volutpat condimentum velit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nam nec ante.

Sed lacinia, urna non tincidunt mattis, tortor neque adipiscing diam, a cursus

ipsum ante quis turpis. Nulla facilisi. Ut fringilla. Suspendisse potenti. Nunc feugiat mi a tellus consequat imperdiet. Vestibulum sapien. Proin quam. Etiam ultrices. Suspendisse in justo eu magna luctus suscipit. Sed lectus. Integer euismod lacus luctus magna. Quisque cursus, metus vitae pharetra auctor, sem massa mattis sem, at interdum magna augue eget diam. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Morbi lacinia molestie dui. Praesent blandit dolor.

Sed non quam. In vel mi sit amet augue congue

Figure 2-3. Mock newsletter using Swiss-style typography

In addition to its focus on simple, sans-serif typefaces, another key element of Swiss design with regard to typography is the use of contrasting font sizes and weights to draw attention to certain points in the text or to create emphasis. This calls for stark differences in font sizes when different font sizes are used, so while some design schools may advocate 12-point headlines and 10-point body text, Swiss design may call for 18-point headlines and 10-point body text, to ensure that there is no question regarding the difference between the two text elements.

Photography

Swiss design style is also marked by the idea that the design should convey a sense of reality and that visual elements will be perceived as “more real” when photographs are used in place of drawn illustrations.

Figure 2-4 shows the sunset over a body of water. The photograph captures the ripples in the water and the effect of the sun’s light on the water in a way that feels very real to the viewer.



Figure 2-4. *Photograph of sunset over water*

Figure 2-5 also depicts a sunset over a body of water. Many of the same elements that are featured in the photograph are present, such as the sun’s reflection over ripples in the water and silhouetted figures, but the theories driving Swiss design hold that viewers are not left feeling as though what they are viewing is real when illustration is used instead of photography. Both the photograph and the painting are pleasing to the eye, but the photograph is more in line with the Swiss style.



Figure 2-5. *Painting of sunset over water*

Iconography

While photographs are preferred to drawings or other illustrations, in many cases, works created using Swiss design often feature the extensive use of icons, either to augment or replace text. This is particularly the case when Swiss design is used in a setting where information must be conveyed to an international audience or in one where you cannot be sure that the viewer in need of the information being conveyed can understand the printed words, regardless of the language in which they are written. Rich iconography used in conjunction with other elements of Swiss design made a big show on the international stage during the 1972 Summer Olympics in Munich, Germany. Otl Aicher designed the brochures and leaflets for the Olympic Games in the Swiss style and used what is now a familiar system of figure icons to represent individuals participating in various events for the games. This facilitated communication with the international audience present for the games. Additional places where you see prominent examples of Swiss design and iconography are bus and train stations, public restrooms (Figure 2-6), and warning labels on many consumer goods.



Figure 2-6. *Familiar Swiss-style design helps to avoid an embarrassing mistake*

Generous Use of Whitespace

In Swiss design, content is king. Too much of anything packed haphazardly into a space is considered excessively cluttered or noisy and a distraction from the information being conveyed. This leads to a design goal that includes plenty of whitespace, to ensure that anything appearing within that expanse will immediately become the focus of attention.

Figure 2-7 shows a dog that appears to be standing watch in a snowy country setting. “The Sentinel” is a descriptive caption, but no particular attention is drawn to either the dog or the caption because the contents are all allowed to run together without any separation and because the trees produce “noise” that detracts from the message of the caption. While this figure is visually appealing, it lacks the stark contrast favored by Swiss design principles. I’ll use the natural whitespace present in the expanse of snow to highlight both the portion of the photo where I want attention focused and the caption, as shown in Figure 2-8.



Figure 2-7. Photo and caption with no whitespace



Figure 2-8. Photo and caption with whitespace for contrast

In Figure 2-8, the only change that I made was to move the text out of the noise produced by the trees, allowing the caption to sit by itself within uninterrupted whitespace. This narrows the focus of the photograph to exclude what is not directly related to the subject and really makes the caption stand out. More of the photograph could have been cropped from the top and bottom to bring even more focus to the subject, but in this case, enough was left to ensure the winter scene did not escape the viewer. Neither the first nor the second version should be considered better or worse, because there are instances when the intent would be to focus on the entire setting and where adhering to the principles of Swiss design is not the goal, in which case, the first treatment illustrated may be preferred.

Strict Organization

In keeping with the overarching theme of clean simplicity and avoidance of anything that distracts from the content, Swiss design is typically marked by strict organization. This is observed in the uniformity of geometric figures as well as in the use of font size to communicate informational hierarchy within text and in the adherence to a grid system to lay out both text and other visual elements in a structured manner. The use of grids is definitely not limited to the Swiss style and has been used in typography design for centuries. With a grid-based design, the design surface is divided into one or more grids, which are used to position text and elements with cells. This provides for an organized and aligned look. At times, the use of grid layout may not be quite as pronounced, because the grid lines need not be perpendicular and parallel with the edges of the design surface, making it possible for a design to follow a grid layout while the content appears angled to the viewer.

Figure 2-9 shows the structural organization achieved by using a grid layout, but it also demonstrates the way that typography is used to achieve organization within the Swiss design style by using a stark difference in font size to delineate different levels within the informational hierarchy. At the highest level of the informational hierarchy, the page header is presented in a 56-point font size. At the next level, group headers are given around one-half the font size of the page header. At the lowest level of the hierarchy for this page, the item title is about half the size of the group header.

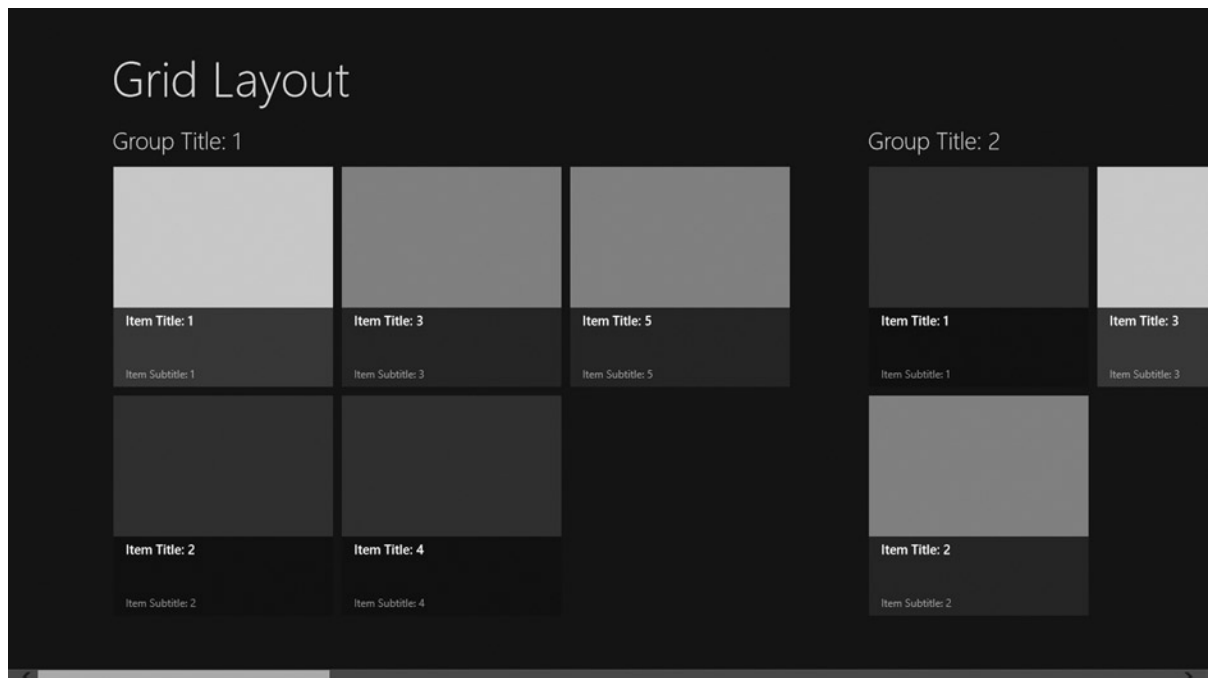


Figure 2-9. Windows Store application demonstrating grid layout and hierarchy

Microsoft Design Language

Rooted heavily in the Swiss design style that I’ve just covered, the Microsoft design language guides user-experience design for the Windows Phone 7/7.5/8 and Windows 8 operating systems as well as for current incarnations of the Zune and Xbox 360 user interfaces, striving to give a consistent look and feel regardless of the device with which you are interacting.

Microsoft Design Language Principles

Microsoft’s earliest guidance on the Microsoft design language characterized it as a confluence of five guiding principles rather than a book of rules or recipes. In this section, I’ll cover the principles that you should weigh when making design choices.

Show Pride in Craftsmanship

Not even the smallest detail should be left to chance in your user interface. Everything the user sees and experiences should be part of the plan and work according to that plan. Additionally, information should be presented according to a carefully thought out visual hierarchy and should be laid out using a grid-based design.

Be Fast and Fluid

Applications should allow users to interact directly with the content and should remain constantly responsive by using motion to provide feedback to interactions. Applications should typically be designed with “touch-first” in mind.

Be Authentically Digital

One of the most flagrant examples of a failed user-experience experiment from Microsoft resulted from the release of Microsoft Bob in 1995. This application was a shell for the operating system that intended to abstract away the whole “computerness” of the computer by providing real-world analogies for different operations. If you wanted to retrieve documents, you clicked the file cabinet. Need to write a letter? Click the pen on the desk! Bob’s failure was driven ultimately by two factors. The first was that it was perceived as childish and patronizing (many shells similar to Bob do find favor in preschool classrooms). The second was that it simply was not an effective way for people to interact with the computer, and introducing abstractions intended to hide the computer tended to make interactions much less efficient, especially for people who have to use a computer for most of the day. The Microsoft design language principles acknowledge that people know they are interacting with a computer and call on designers to embrace the medium. This includes using the cloud to keep users and apps connected and effectively using motion and bold, vibrant colors to communicate with the user.

Do More with Less

Windows 8 provides rich functionality to allow applications running both on your device and in the cloud to interact with each other. This enables applications to focus on doing a very narrowly defined set of things and to do one thing in an extraordinary manner rather than do several things poorly. In keeping with the Bauhaus and Swiss design influences, the content should be the primary focus of attention, and very little else should be present to distract from this content. The full-screen nature of Windows Store apps even removes the need for window chrome, allowing a completely immersive experience, so that when the user is in your application, your application receives all of his or her attention.

Win As One

One of the keys to working in a Windows Store application is that the style has been set. Users of a Windows Store application will be opening your application with the expectation that they will already have some degree of familiarity with it, because they are familiar with the look and feel of other Windows Store applications. Some things

that can really be harmful to individual applications and, eventually, to the ecosystem in which the applications reside are design decisions that radically change the design paradigm of the application to give users something “new” and “better” than what they are used to having. You should strive to impress your users with how well your application does the things it is meant to be good at, but trying to surprise those users by changing user-interface and navigation paradigms will only confuse them and make them lose trust in your application. Microsoft has provided guidance, tools, templates, and style sheets to make it easy for developers to create Windows Store applications with a consistent look and feel, and you should make full use of these resources.

User-Experience Guidelines for Windows Store Apps

In addition to the more generalized principles that Microsoft has published for Windows Store applications, a comprehensive set of guidelines has also been made available in order to provide detailed prescriptive guidance in regard to the look, feel, and behavior of applications designed to run in this new ecosystem. Although not a comprehensive treatment of these guidelines, which are freely available in their entirety on the MSDN Library web site at <http://dev.windows.com>, this section covers a few of the aspects that are most applicable to designers/developers getting a feel for the Windows 8 experience.

Application Layout

Applications should be designed using a grid layout, organized using either a hierarchical navigation scheme or a flat view, as dictated by the content.

When a hierarchical approach is taken, the top of the hierarchy represents the lowest level of detail, and each subsequent level in the navigation hierarchy zooms in with increasing detail. Typically, the highest level, sometimes referred to as the *Hub*, is the entry point of the application and reveals one or more groups that the user can drill into (see Figure 2-10).

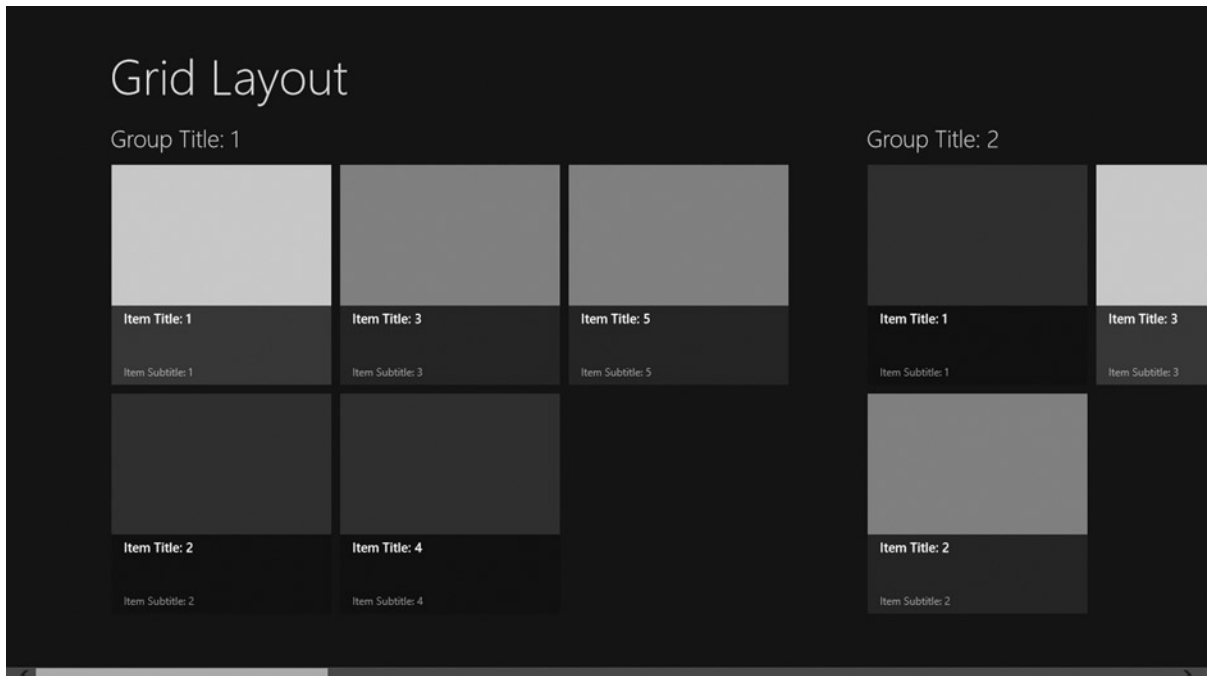


Figure 2-10. Hierarchical navigation at highest level (the Hub)

By selecting a group from the main Hub, the next level of navigation (commonly referred to as a *Section*) is revealed. The Section page is arranged to provide some context about the Section itself and lists the individual items that are the lowest level of navigation and highest level of detail (see Figure 2-11).

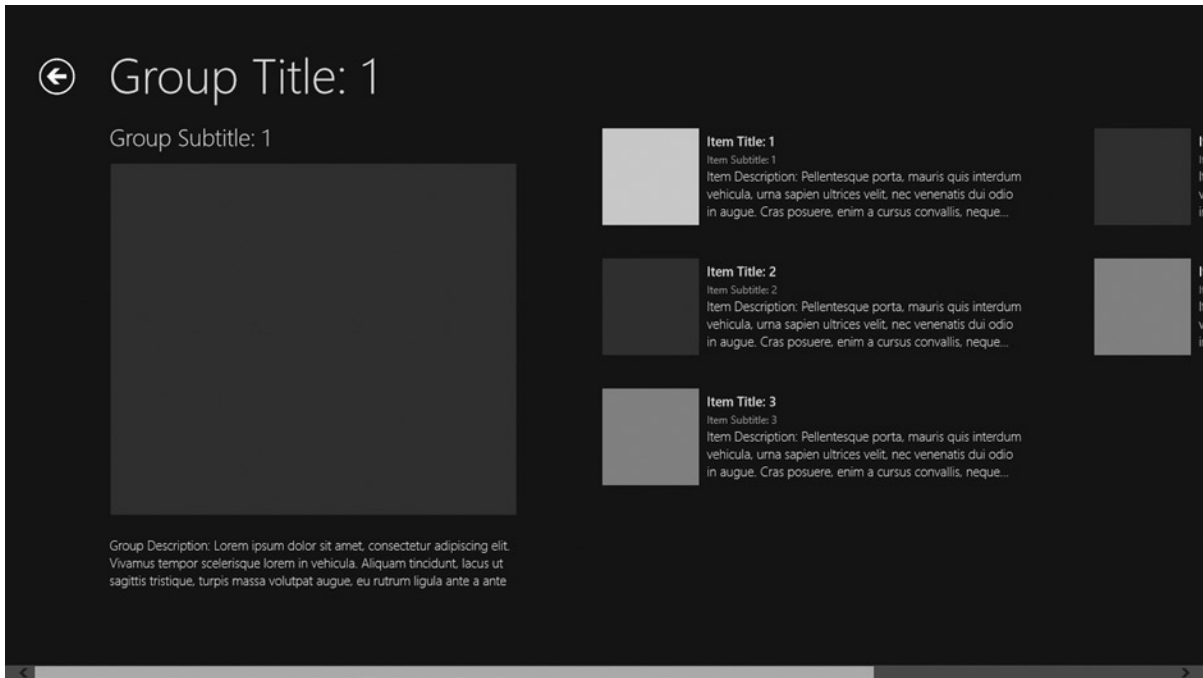


Figure 2-11. Hierarchical navigation at Section level

From the Section page, the user is offered a way to navigate back up a level, typically through the use of a back arrow, as shown in Figure 2-11, to the Hub, a means to navigate to sibling Section pages through a swipe gesture (if touch enabled) or through the use of arrows at the left and right edge of the screen centered vertically, or items to select in order to continue to the Detail page. At the Detail page level of navigation, a granular view of the item data is presented (see Figure 2-12). As with the Section page, the back arrow is presented to allow for navigation up the hierarchy to the Section page in which the item is organized. As with Section pages, users can choose to navigate between Detail pages within the same section through the use of a swipe gesture on touch-enabled systems or through interaction with arrows at the left and right edge of the screen. The hierarchical navigation is especially well suited for browsing and interacting with information that can be fit into master-detail categorization.



Figure 2-12. Hierarchical navigation at Detail page

Many applications do not fit into the master-detail categorization that works well with a hierarchical navigation structure and focus more on the document-based style familiar with Microsoft Word, Excel, or Internet Explorer. For this type of application, a flat navigation system works much better. At the core of the flat navigation is that content is separated into pages with information that is either unrelated or at the same hierarchical level (see Figure 2-13). The navigation bar is presented when activated by the user and is employed to switch between active documents, often presenting a command that the user can access to add a document to the session (see Figure 2-14).

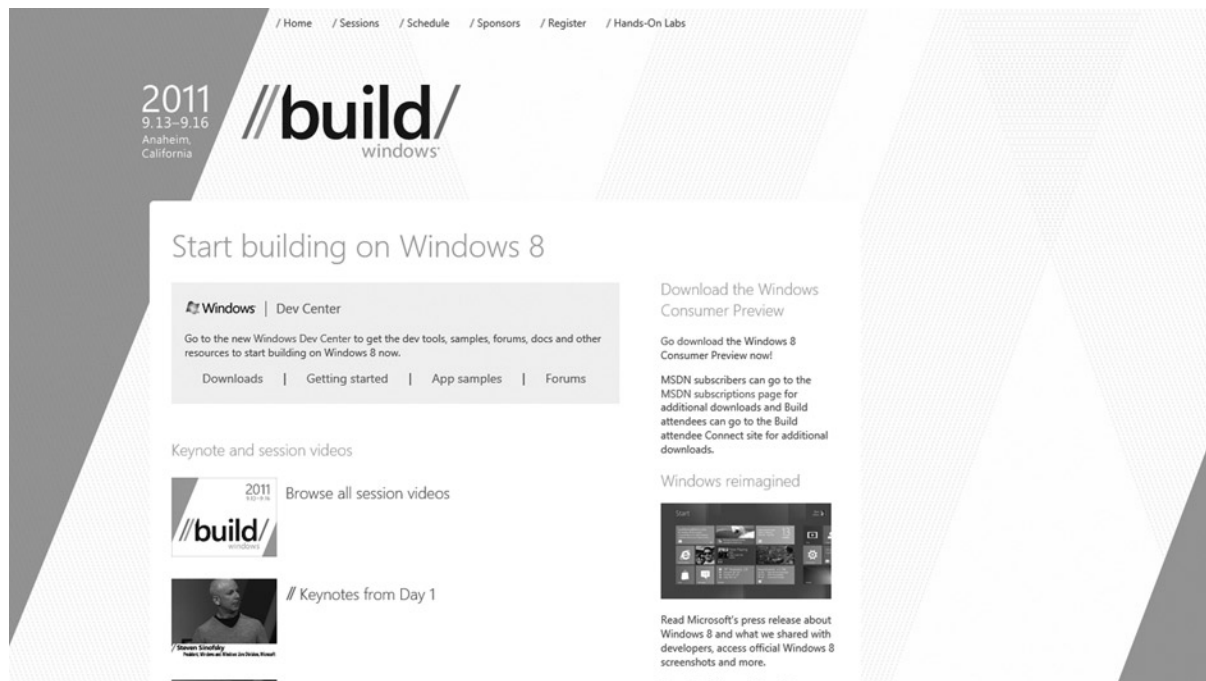


Figure 2-13. Internet Explorer's design presents a flat view with a single document using an entire viewport

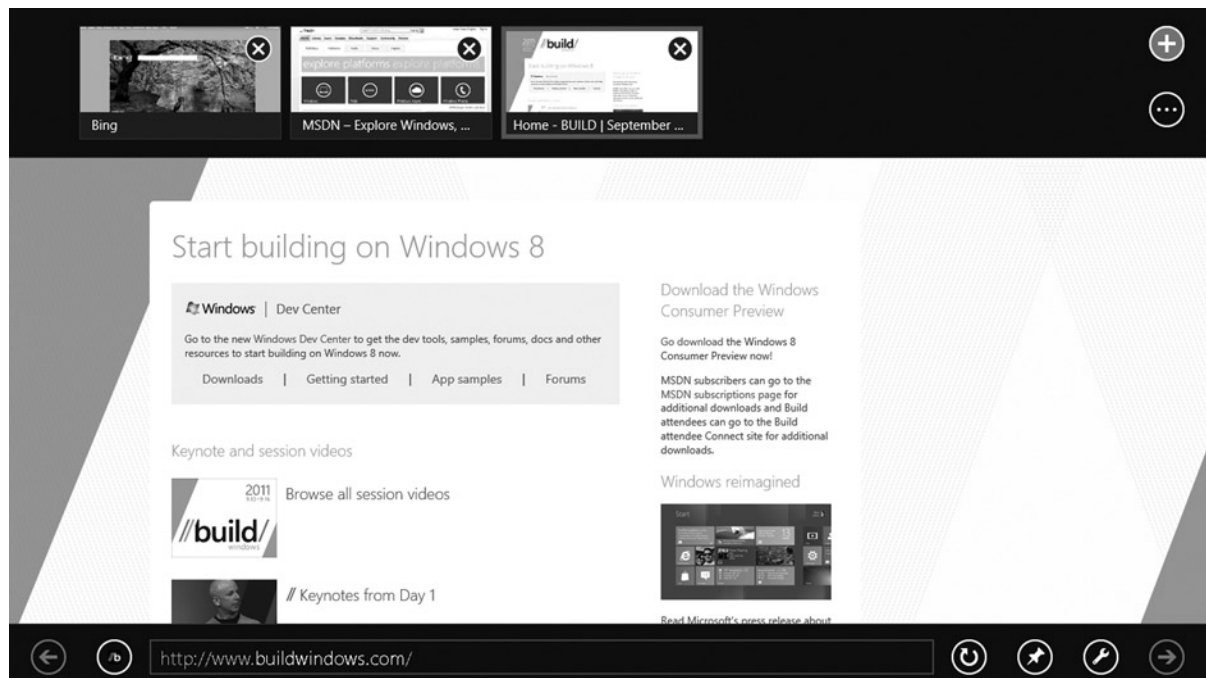


Figure 2-14. Internet Explorer with navigation bar activated to switch active document

Typography

With its heavy emphasis on typography and text-centered content, no coverage of the user-experience guidelines for Windows Store applications would be complete without providing advice for the formatting and use of text. Following in the tradition of Swiss design, consistent fonts should be used when building applications. Which specific font should be used varies according to the purpose of the text. Text that is intended to be used for buttons or labels on UI elements should favor the Segoe UI font, which is used throughout Windows 8 user-interface elements (see Figure 2-15).



Figure 2-15. Segoe UI is used for labels and other UI elements

Blocks of text that are to be presented to the reader in a read-only fashion, such as news articles, should favor the serif Cambria font, because readers are accustomed to extended blocks of text being presented in a serif font (see Figure 2-16). This font should be presented in 9 points, 11 points, or 20 points, depending on the need to draw focus or show emphasis. This is a departure from the Swiss style's preference for sans-serif fonts in all things, because the Microsoft design team found serif fonts to be easier on the eyes for extended reading.

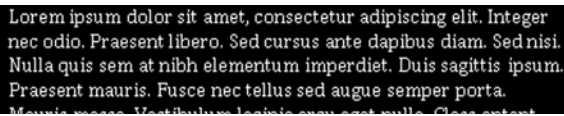


Figure 2-16. Cambria for read-only text blocks

Continuous blocks that are intended for the user to both read and edit should favor the sans-serif font Calibri (see Figure 2-17). The recommended size for this font is 13 points, which shares the same height as 11-point Segoe UI, so the two will maintain a consistent appearance when used together on the same line.

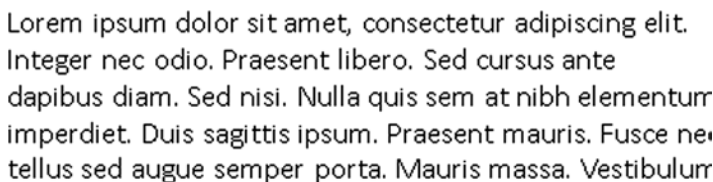


Figure 2-17. Calibri for read-edit text blocks

Regardless of the font face, when emphasis is needed on certain pieces of text, the appropriate way to produce emphasis is through the use of stark contrast with the font size or the font weight. At the same level within the information hierarchy, weight is used for emphasis, while size draws the distinction between levels. Using text decorations such as underline or italics reduces clarity and should not be used for emphasis in a Windows Store application.