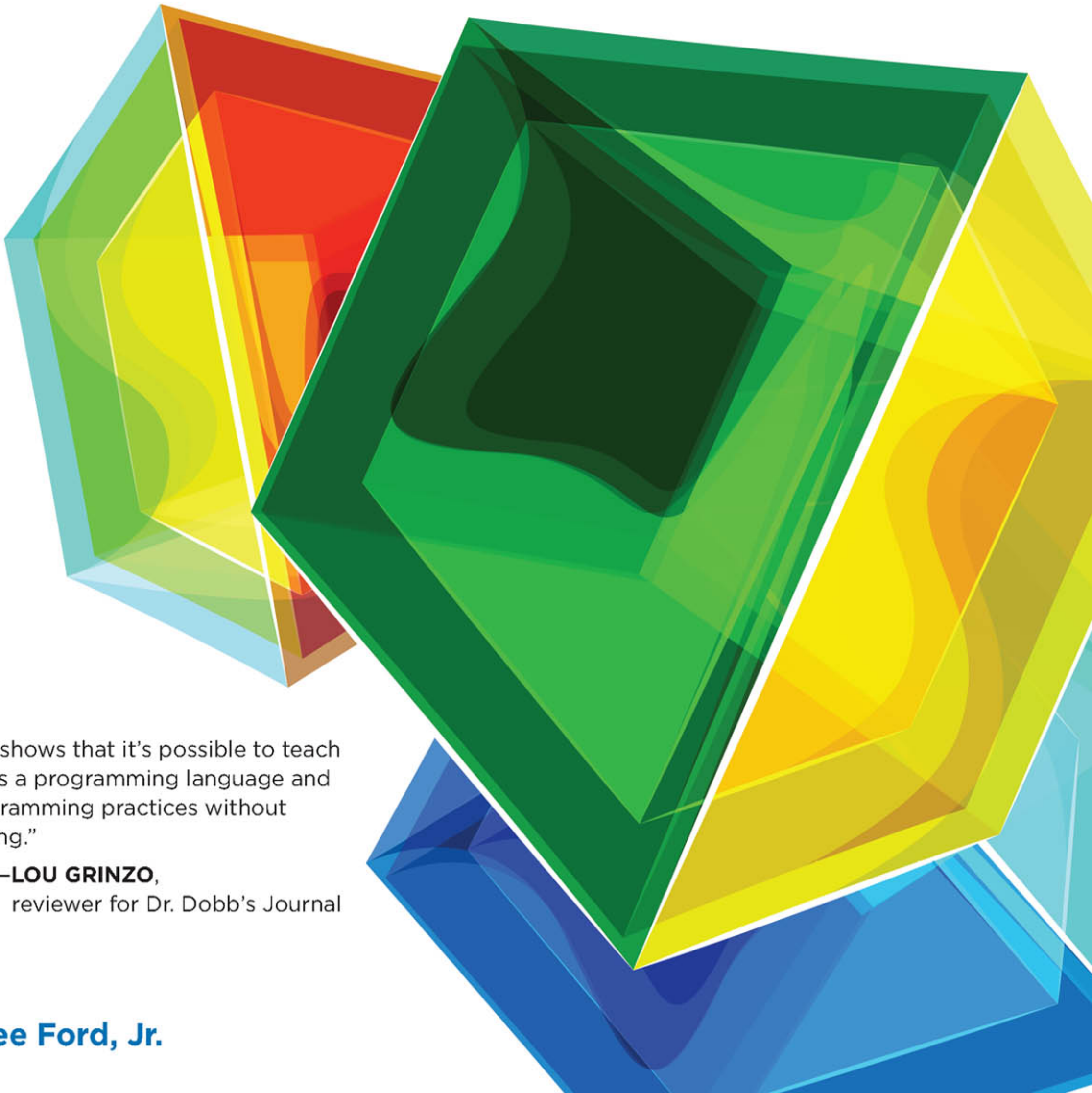
Web Downloads Available

Microsoft®

WSH and VBScript Programming for the Absolute Beginner

Fourth Edition



"This series shows that it's possible to teach newcomers a programming language and good programming practices without being boring."

—**LOU GRINZO**, reviewer for Dr. Dobb's Journal

Jerry Lee Ford, Jr.

Microsoft® WSH and VBScript Programming for the Absolute Beginner, Fourth Edition

Jerry Lee Ford, Jr.

Cengage Learning PTR

CENGAGE Learning®
Professional • Technical • Reference

Australia, Brazil, Japan, Korea, Mexico, Singapore, Spain, United Kingdom, United States

CENGAGE Learning®

Microsoft® WSH and VBScript Programming for the Absolute Beginner, Fourth Edition
Jerry Lee Ford, Jr.

Publisher and General Manager, Cengage Learning PTR:
Stacy L. Hiquet

Associate Director of Marketing:
Sarah Panella

Manager of Editorial Services:
Heather Talbot

Senior Marketing Manager:
Mark Hughes

Senior Acquisitions Editor:
Mitzi Koontz

Project and Copy Editor:
Kate Shoup

Technical Reviewer:
Zac Hester

Interior Layout:
Shawn Morningstar

Cover Designer:
Mike Tanamachi

Indexer:
Larry Sweazy

Proofreader:
Sam Garvey

Library of Congress Control Number: 2014933749

ISBN-13: 978-1-305-26032-0

ISBN-10: 1-305-26032-5

eISBN-10: 1-305-26033-3

Cengage Learning PTR
20 Channel Center Street
Boston, MA 02210
USA

Printed in the United States of America
1 2 3 4 5 6 7 16 15 14

To my father;
to my children, Alexander, William, and Molly;
and to my beautiful wife, Mary.

Acknowledgments

There are a number of individuals to whom I owe many thanks for their help and assistance in the development of the fourth edition of this book. I would start by thanking Mitzi Koontz, who served as the book's acquisitions editor. Special thanks also go out to Kate Shoup for serving as the book's project editor and copy editor. I also want to thank Zac Hester for all his valuable technical input and advice. In addition, I would like to thank everyone else at Cengage Learning for all their hard work.

About the Author

Jerry Lee Ford, Jr. is an author, educator, and IT professional with more than 24 years of experience in information technology, including roles as an automation analyst, technical manager, technical support analyst, automation engineer, and security analyst. He is the author of 38 books and co-author of two additional books. His published works include *Microsoft Windows PowerShell Programming for the Absolute Beginner*; *Microsoft Visual Basic 2008 Express Programming for the Absolute Beginner*; *HTML, XHTML, and CSS for the Absolute Beginner*; *XNA 3.1 Game Development for Teens*; and *VBScript Professional Projects*. Jerry has a master's degree in business administration from Virginia Commonwealth University in Richmond, Virginia, and has more than five years of experience as an adjunct instructor teaching networking courses in information technology.

Table of Contents

This page intentionally left blank

Introduction

Welcome to the fourth edition of *Microsoft WSH and VBScript Programming for the Absolute Beginner*. Visual Basic Scripting language (VBScript) is a member of the Visual Basic family of programming languages. Other members of this family include Visual Basic and Visual Basic for Applications (VBA). Visual Basic is a very powerful and complex programming language used by programming professionals all over the world. VBA is a programming language based on Visual Basic that is designed to provide a programming environment for Microsoft Office applications such as Excel and Access.

Like VBA, VBScript represents a subset of the Visual Basic programming language. VBScripts can be run on any computer running Windows 95 or later as long as the Windows Script Host (WSH) is installed. The WSH represents one of several environments in which VBScripts can be run. Other environments in which VBScripts can run include HTML pages processed by Internet Explorer–compatible Web browsers and within Microsoft Outlook or Active Server Pages (ASP). Of all the environments in which VBScripts can run, the WSH is the most commonly used. However, by learning to write VBScripts using the WSH, you are also learning much of the prerequisite knowledge required to write VBScripts that will run in each of these other environments.

The WSH provides VBScripts with the capability to execute on Windows computers and to directly access and manipulate Windows resources such as the Windows desktop, file system, Registry, printers, network resources, and so on. You can think of the relationship between VBScript and the WSH as follows: VBScript provides the capability to create scripts and apply logic to perform specific tasks that manipulate Windows resources, which are made available to the script via the WSH.

Why VBScript?

VBScript is an excellent first programming language to learn. Its simplicity makes learning basic programming concepts easy. Yet VBScript is a powerful scripting language from which you can learn even the most complex programming concepts such as how to perform object-based programming. Unlike Visual Basic, VBA, and many other programming languages, there is no complex development environment to learn. In fact, you can create all your VBScripts using a simple text editor such as Windows Notepad.

VBScript provides a foundation that will later make learning Visual Basic and VBA a lot easier. VBScript is a great language for developing small but powerful scripts that perform all sorts of tasks. In fact, you'll find that many VBScripts are not very big at all when compared to programs written using more traditional programming languages. As you read through this book, I think you will be amazed at just what you can do with only a handful of lines of VBScript code. This makes VBScript the perfect language for rapid development, meaning that you can often write a VBScript to perform a task in a fraction of the time that it might take to write a program that performs the same task using a different programming language. Best of all, VBScript is free.

Who Should Read This Book?

This book is designed to teach you how to begin developing VBScripts using the WSH. It does not assume that you have a programming background. However, a basic understanding of computers and Microsoft Windows is assumed.

If you are a first timer looking for a friendly language with which to begin a programming career or a more experienced programmer who is looking for a book that provides you with a quick WSH and VBScript learning curve, then give this book a try.

This book's games-based teaching approach makes it very different from other books. This approach is not only more fun, but is also an extremely helpful technique for learning a new programming language.

What You Need to Begin

To follow along and complete all the exercises that you'll find in this book, you'll need a number of things:

- A computer running Windows.
- The current version of the WSH, which is version 5.8. If your computer is running Windows 7, Windows 8, or Windows 8.1, then you already have the version of WSH that you need. If you are using Windows XP with Service Pack 3 or Windows Vista, you can download and install WSH 5.7 from www.microsoft.com/downloads/.
- A text editor that supports the creation of plain-text files to create and work with your VBScripts. For this book, you can use the Windows Notepad application. Alternatively, you may prefer to download and install a VBScript-compatible script editor. Specialized VBScript editors provide numerous advanced features not provided by Notepad, including statement color-coding, a built-in debugger, line and column numbering, script execution from within the editor, statement indentation, and more. A good example of a VBScript editor is Adersoft VbsEdit. It is distributed as shareware with a limited period of free trial and can be downloaded from www.vbsedit.com.

How This Book Is Organized

The fourth edition of *Microsoft WSH and VBScript Programming for the Absolute Beginner* has been improved in a number of ways. For starters, it has been updated to cover WSH 5.8 and VBScript 5.8, both of which were updated with the release of Windows 7. All scripts have been tested and their execution verified on both Windows 7 and Windows 8.1. In addition, a new chapter has been added that provides an introduction to HTML Applications (HTAs), which provide a mechanism for creating scripts that feature a graphical user interface (GUI). Lastly, I have streamlined coverage of many topics spread throughout the book to provide an even better learning experience.

This book is organized into four parts with the intention that you read it sequentially from beginning to end. If you are a new or inexperienced programmer, you will want to read this book in this manner. However, if you already know another programming language and feel that you have a strong enough background in basic programming concepts, you might want to skip around and tackle each chapter in the order that best suits your particular requirements.

Part I, "Introducing the WSH and VBScript," consists of two chapters and provides an introduction to both VBScript and the WSH. Part II, "Learning VBScript and WSH Scripting," contains five chapters, which cover the programming statements that make up the VBScript scripting language. In addition, you'll find coverage of the WSH woven throughout these chapters. The seven chapters in Part III, "Advanced Topics," are dedicated to covering a collection of advanced topics that include file and folder administration, error handling, interaction with the Windows Registry, working with built-in VBScript objects, using XML to create WSH files, working with Windows Management Instrumentation, and adding graphical user interfaces to your scripts. Part IV, "Appendices," is a collection of five appendices that provide you with additional avenues of exploration, including examples of real-world scripts, an introduction to Remote WSH, documentation of built-in VBScript functions, and a look at this book's companion website.

The basic outline of the book is as follows:

- **Chapter 1, "Getting Started with the WSH and VBScript."** This chapter provides a high-level introduction to both the WSH and VBScript. This includes how to install the WSH and how to create and execute your first VBScript.
- **Chapter 2, "An Introduction to the Windows Script Host."** This chapter provides an introduction to the WSH core object model and the objects that comprise it. Particular attention is paid to the `WScript` root object. You'll also learn how to configure the WSH and how to specify a default script execution host.
- **Chapter 3, "VBScript Basics."** This chapter begins your VBScript education. You'll learn about VBScript's core and run-time objects and their properties and methods. You'll learn about other VBScript elements including VBScript's built-in functions, syntax rules, and output methods. You'll also learn about various WSH output functions.
- **Chapter 4, "Constants, Variables, Arrays, and Dictionaries."** This chapter shows you how to create and reference data stored in the computer's memory using constants, variables, and arrays. You'll learn about VBScript's built-in collection constants. This chapter also presents the rules for variable creation and the enforcement of variable use as well as the techniques required to store and retrieve collections of data in arrays.
- **Chapter 5, "Conditional Logic."** This chapter expands your scripting background to include an understanding of how to add conditional logic to your scripts to provide alternative execution paths for script execution. You'll examine both the VBScript `If` and `Select Case` statements. In addition, you'll learn about VBScript operators and operator precedence.
- **Chapter 6, "Processing Collections of Data."** This chapter teaches you how to process collections of data and resources using various VBScript looping statements (`For…Next`, `Do While`, `Do…Until`, `While…End`, and `For Each…Next`). You'll learn how to write small scripts that can add shortcuts to your scripts on the Windows desktop and Start menu.

- **Chapter 7, "Using Procedures to Organize Scripts."** In this chapter, you learn how to improve the organization of your scripts using procedures. You'll also be introduced to the concept of creating reusable procedures. This will help you create scripts that are more complicated and easier to modify.
- **Chapter 8, "Storing and Retrieving Data."** This chapter teaches you how to create VBScripts that can write to and read from text files. In addition to learning how to create reports and log files, this chapter shows you how to store and retrieve script configuration settings in INI files, thus allowing you to externalize key script settings.
- **Chapter 9, "Handling Script Errors."** This chapter focuses on teaching you how to deal with the errors that occur during script development and execution. This chapter introduces errors during script development and shows you how to troubleshoot them. In addition, you'll learn how to bypass errors and to develop code that handles specific error conditions.
- **Chapter 10, "Using the Windows Registry to Configure Script Settings."** This chapter provides an overview of the Windows Registry and shows you how to develop scripts that store and retrieve data in Registry keys and values. Because most Windows functionality is controlled from the Registry, this knowledge will provide the basic building blocks required to manipulate any number of Windows settings.
- **Chapter 11, "Working with Built-in VBScript Objects."** This chapter expands your understanding of object-based programming by reviewing VBScript's built-in collection of objects. Specifically, you'll learn new techniques for parsing and extracting data from strings.
- **Chapter 12, "Combining Different Scripting Languages."** In this chapter, you learn how to take advantage of the WSH's support for Windows Script Files. Windows Script Files enable you to combine two or more WSH-supported scripting languages, such as VBScript and JScript, into a single script using XML. You'll also learn a little about XML and the XML tags supported by the WSH.
- **Chapter 13, "Working with the Windows Management Instrumentation."** This chapter was added to the third edition of this book. It provides an overview of the WMI and the WMI model. You will learn about WMI objects, namespaces, providers, and classes. You will also learn how to develop scripts that use WMI to collect and process systems information.
- **Chapter 14, "Adding a GUI to Your Scripts."** This chapter, which is entirely new to this edition of the book, introduces HTML Applications (HTAs), which can be used to provide scripts with graphical user interfaces (GUIs). This chapter explains how HTAs work and teaches you how to add GUIs to your scripts, replete with radio buttons, checkboxes, text fields, drop-down lists, and other attributes associated with Windows applications.
- **Appendix A, "WSH Administrative Scripting."** In this appendix, I show you some practical examples that demonstrate the use of VBScript and the WSH in real-world situations. This appendix will assist you in making a transition from the book's game-based approach to real-world script development.
- **Appendix B, "Introducing Remote WSH."** This appendix was added to the third edition of this book. In it, you will learn how to execute, monitor, and terminate the remote execution of scripts using Remote WSH. You will learn about the objects that make up Remote WSH and how to work with their properties and methods.

- **Appendix C, "The WSH Core Object Model."** This appendix provides detailed information on the WSH core object model using material previously presented in Chapter 2. This includes a detailed examination of WSH objects' methods and properties.
- **Appendix D, "Built-in VBScript Functions."** In this appendix, I list and define all the functions that are available as you develop your VBScripts.
- **Appendix E, "What's on the Companion Website?"** In this appendix, I provide more information about the sample scripts provided on the book's companion website (www.cengageptr.com/downloads).

Conventions Used in This Book

To help make this book as easy as possible to read and understand, a number of conventions have been applied to help highlight critical information and to emphasize specific points. These conventions are as follows:

Hint

Whenever I can, I provide tips on how to do things differently and point out techniques that you can try to become a better programmer in "hint" boxes.

Trap

From time to time, I use "trap" boxes to point out areas where you are likely to run into problems and then provide you with advice on how to deal with these situations—or, better yet, to prevent them from happening in the first place.

Trick

Whenever I can, I share programming shortcuts that will help to make you a better and more efficient programmer. These appear in "trick" boxes.

In the Real World

Throughout the book, I'll stop along the way to point out how the knowledge and techniques that you are learning can be applied to real-world scripting projects. These will appear in "real world" boxes.

Definition

Whenever a new term is introduced, I will provide you with an explanation of that term's meaning in a "definition" box.

In addition, toward the end of each chapter, you will find instructions that guide you through the development of a new computer game. In most chapters, immediately following each game project, you will find a series of suggestions or challenges designed to provide you with ideas that you should be able to apply in order improve the game and further the development of your programming skills. These appear under a "Challenge" heading.

Companion Website Downloads

You may download the files for this book from www.cengageptr.com/downloads. For more information about what files are available, see Appendix E.

PART

I Introducing the WSH and VBScript

This page intentionally left blank

Getting Started with the WSH and VBScript

In this chapter, you'll be introduced to a number of topics. These include a high-level overview of the Windows Script Host (WSH) and VBScript. You will learn how the WSH and VBScript work together to provide a comprehensive scripting environment. You will also be introduced to HTML Applications (HTAs) and learn how an HTA can be used to provide your scripts with a graphical user interface (GUI). In addition, you'll learn a little bit about VBScript's history and its relationship to other languages in the Visual Basic programming family. As a wrap-up, you'll learn how to create and execute your very first VBScript.

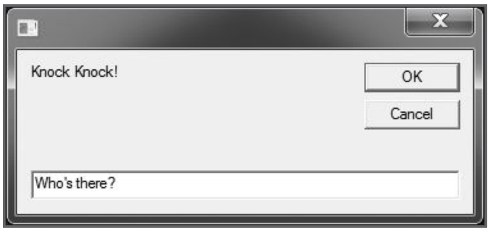
Specifically, you will learn the following:

- The basic mechanics of the WSH
- How to write and execute VBScripts using the WSH
- Background information about VBScript and its capabilities
- How you can use HTAs to add GUIs to your scripts
- How to create your first VBScript game

Project Preview: The Knock Knock Game

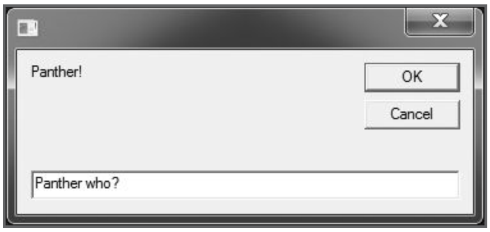
In this chapter, as in all the chapters to follow, you will learn how to create a computer game using VBScript. This chapter's game is called the Knock Knock game. Actually, it's more of a riddle than a game, but it provides a great starting point for demonstrating how VBScript works and how it can be used to develop games and other useful scripts.

The Knock Knock game begins by displaying a pop-up dialog box that reads "Knock Knock." It then waits for the user to respond with "Who's there?" The dialog between the game and the player continues until the computer finally displays the game's punch line. Figures 1.1 through 1.3 demonstrate operations of the script on Windows 7 and show the flow of the conversation between the game and the player. Figure 1.4 shows the message that appears if the player does not play the game correctly.


Knock Knock!

Who's there?

Figure 1.1 The game begins by knocking on the door and waiting for the player to respond. © 2014 Cengage Learning.


Panther!

Panther who?

Figure 1.2 The first clue is provided.
© 2014 Cengage Learning.

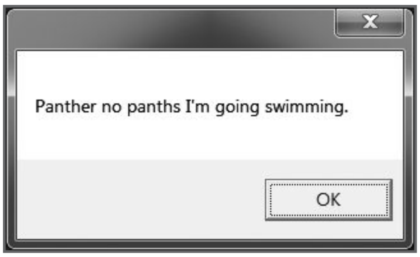

Panther no panths I'm going swimming.

Figure 1.3 The joke's punch line is delivered.
© 2014 Cengage Learning.

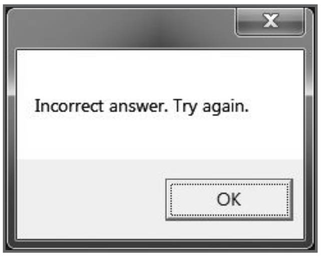

Incorrect answer. Try again.

Figure 1.4 If the user makes a mistake when playing the game, an error message providing another invitation to play the game appears. © 2014 Cengage Learning.

By the time you have created and run this game, you'll have learned the fundamental steps involved in writing and executing VBScripts. At the same time, you will have prepared yourself for the more advanced programming concepts developed in later chapters, including how to use the WSH and VBScript to develop some really cool games.

What Is the WSH?

The Windows Script Host (WSH) is a programming environment that allows you to write and execute scripts that run on Windows operating systems. You can use the WSH to create and execute *scripts*—small text-based files written in an English-like programming language—from the Windows command prompt or directly from the Windows desktop. Scripts provide quick and easy ways to automate lengthy or mundane tasks that take too much time or effort using the Windows graphical user interface (GUI). Scripts are also better suited for automating tasks that are not complex enough to justify the development of an entire application using a language such as C++ or Visual Basic.

The WSH is made up of a number of different components. These components include the following:

- Script engines
- Script execution hosts
- The WSH core object model

The relationship of each of the components to one another is shown in Figure 1.5.

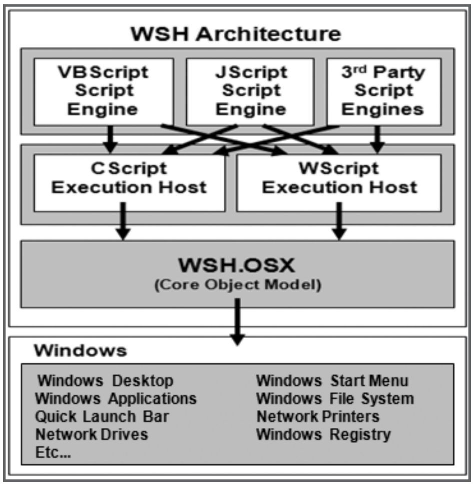


Figure 1.5 The components that comprise the WSH.
© 2014 Cengage Learning.

WSH Scripting Engines

A *script execution engine* is a program that processes (interprets) the statements that make up scripts and translates them into machine-readable code that the computer can understand and execute. By creating an environment in which scripts can execute, the WSH makes script development a straightforward task.

The WSH provides each script with a number of resources. The WSH provides script engines for processing scripts. By default, Microsoft provides two script engines for the WSH:

- **VBScript.** A scripting language based on Microsoft's Visual Basic programming language.
- **JScript.** A scripting language based on Netscape's JavaScript Web-scripting language.

Therefore, by default, the WSH can process scripts written in either VBScript or JScript. The WSH is designed in a modular fashion, allowing Microsoft and third-party software developers to add support for additional scripting engines. For example, script execution engines have been developed for Perl, Python, and Rexx.

Selecting a WSH Script Execution Host

To actually run a script, the WSH uses a script execution host to process a script after a script engine has interpreted that script. The WSH supplies two different script execution hosts:

- **CScript.exe.** An execution host that enables scripts to execute from the Windows command prompt and display text-based messages.
- **WScript.exe.** An execution host that enables scripts to execute from the Windows desktop, display messages, and collect user input using graphical pop-up dialog boxes.

With the exception of the WScript.exe execution host's capability to display graphical pop-up dialog boxes, the functionality provided by the WSH's two execution hosts is identical. In fact, if you run a script using the CScript.exe execution host, the script can, depending on how it is written, still display messages using pop-up dialog boxes.

Definition

Within the context of this discussion, the term *host* describes an environment that provides all the resources required for a script to execute.

As both execution hosts provide the same basic functionality, you're probably wondering which one you should use. There's no right or wrong answer here. Often, the selection of an execution host is simply a matter of personal preference. However, there are some circumstances in which you may want to choose one over the other. For example, if you plan to run your scripts in the background, or if you want to schedule the execution of your scripts using the Windows Task Scheduler service and have no requirement for interacting with the user, you might want to use CScript.exe. However, if your scripts need to interact with the user—which will be the case with the games you'll create with this book—you'll want to use the WScript.exe execution host. Another factor that may affect your selection of a script execution host is your personal comfort level in working with the Windows command prompt.

Introducing the WSH Core Object Model

The WSH provides one final component, called the *core object model*, which is critically important to the development and execution of scripts. The WSH core object model provides VBScript with direct access to Windows resources.

Examples of the types of Windows resources to which the WSH core object model provides access include the following:

- Windows desktop
- Windows Start menu
- Windows applications
- Windows file system
- Network printers
- Network drives
- Windows Registry

The Windows operating system can be viewed as a collection of objects. For example, a file is an object. So is a folder, disk drive, printer, or any other resource that is part of the computer. What the core object model does is expose these objects in a format that allows scripts to view, access, and manipulate them. Each exposed object has associated properties and methods that scripts can then use to interact with an object, as well as affect its behavior or status. For example, a file is an object, and a file has a number of associated properties, such as its name and file extension. By exposing the Windows file system, the WSH enables scripts to access files and their properties and to perform actions, such as renaming a particular file or its file extension. Files also have methods associated with them. Examples of these methods are those that perform the copy and move operations. Using these methods, you can write scripts that can move or copy files from one folder to another or, if you are working on a network, from one computer to another.

Definition

In this book, the term *property* refers to an object-specific attribute, such as a file's name, that can be used to affect the status of the object.

Don't worry if the WSH core object model seems a little confusing right now. You will learn more about it in Chapter 2, "An Introduction to the Windows Script Host." In addition, you can jump to Appendix C, "The WSH Core Object Model," at any time for additional insight. The important thing to understand for now is that the WSH enables scripts to access Windows resources (objects) and to change their attributes (properties) or perform actions that affect them (using object methods).

Definition

In this book, the term *method* is used to refer to a built-in function that your scripts can execute to perform an action on an object, such as to copy or move a file to another location.

How Does the WSH Compare to Windows Shell Scripting?

Windows shell scripts are plain text files that have a .bat or .cmd file extension. Unlike scripts written to work with the WSH, which are written using specific scripting languages like VBScript and JScript, Windows shell scripts are developed using regular Windows commands and a collection of shell-scripting statements. The WSH provides a more complete scripting environment due in large part to its core object model. However, Windows shell scripts still offer a powerful scripting solution. This is partly because you can execute any Windows command or command-line utility from within a shell script. Windows shell scripting also provides a complete collection of programming statements that include support for variables, looping, conditional logic, and procedures. For non-programmers, shell scripts may be easier to read, understand, and modify.

Another difference between scripts written using the WSH and Windows shell scripts is that Windows shell scripts only support text-based communications with the user. In other words, shell scripts cannot display messages or prompt the user for information using graphical pop-up dialog boxes. Windows shell scripting does not provide support for any type of object model like the WSH does. Therefore, Windows shell scripts are not capable of directly interacting with many Windows resources. For example, Windows shell scripts cannot directly edit the Windows Registry or create desktop shortcuts. However, Windows Resource Kits provide Windows shell scripts with access to a number of command-line utilities that provide indirect access to many Windows resources.

To write shell scripts, you must have a good understanding of Windows commands and their syntax. You must also be comfortable working with the Windows command prompt. Conversely, to effectively use the WSH, you must be well versed in one of its supported scripting languages. There are many cases in which you can accomplish the same task using either Windows shell scripting or the WSH. As a general rule, however, the more complex the task, the more likely you'll want, or need, to use the WSH.

Definition

A *Windows Resource Kit* is a combination of additional utilities and documentation designed for a particular Windows operating system but provided as a separate downloadable package. You can obtain Windows Resource Kits via the Microsoft Download Center (www.microsoft.com/en-us/download).

Hint

If you're really interested in learning more about Windows shell scripting, read *Microsoft Windows Shell Script Programming for the Absolute Beginner* (ISBN 1-59200-085-1).

WSH Versus Windows PowerShell

PowerShell is fully integrated into Microsoft's .NET framework, providing system administrators with access to system resources. Like VBScript and the WSH, Windows PowerShell is object oriented. PowerShell lets you execute PowerShell commands, referred to as cmdlets, and develop and execute small scripts that use those cmdlets.

Like WSH and VBScript, Windows PowerShell provides access to system resources and can be used to programmatically interact with the files system, Windows Registry, .NET, and WMI. WSH, VBScript, and Windows PowerShell support a robust collection of language constructions like variables, conditional logic, loops, and functions.

Unlike WSH and VBScript, PowerShell does not support the use of pop-up dialog boxes and is restricted to the command line. Unlike VBScript, which is based on the widely popular and easy-to-use BASIC programming language, Windows PowerShell represents a completely new scripting language, which is arguably more difficult for new programmers to learn and understand.

Microsoft has put a lot of time and resources into the development of Windows PowerShell and is promoting it as the future of Windows scripting. However, Microsoft is continuing to support the WSH as a Windows scripting environment, as evidenced by the recent release of WSH 5.8. Microsoft will continue to support the WSH—and for good reason. Companies all over the world have invested significant time and resources in it and have developed hundreds of millions of lines of code that are used to run mission-critical applications and administer servers and workstations.

Companies continue to rely on the WSH and VBScript and extend their use. As such, WSH and VBScript programming will remain essential for application developers and systems administrators for the foreseeable future.

Hint

If you're really interested in learning more about Windows PowerShell, read *Microsoft Windows PowerShell 2.0 Programming for the Absolute Beginner, Second Edition* (ISBN 1-59863-899-8).

Understanding How the Windows Shell Works

Even if you have used Windows operating systems for many years, chances are that you have only limited experience working with the Windows shell. To become a really efficient and proficient script programmer, you'll need a solid understanding of what the Windows shell is and how to work with it.

An understanding of how to work with the Windows shell is also important when learning how to work with the Cscript.exe execution host, because scripts run by this execution host are generally started from the Windows command prompt. Finally, it's important to understand the Windows shell when working with the WScript.exe execution host because it provides support for command-line script execution.

Definition

The Windows *command prompt* enables you to submit commands to the Windows shell for processing. By default, the command prompt appears in the form of a drive letter followed by a colon, the backslash character, and then the greater-than character (for example, C:\>).

You cannot touch the Windows operating system itself. This would be far too complex and difficult. Instead, you must go through an interface. Windows operating systems support two such interfaces:

- **The Windows GUI.** The Windows GUI is provided in the form of the Windows desktop, Start menu, and other graphical elements with which you normally interact when using your computer. The purpose of the GUI is to make the operating system easier to work with.
- **The Windows shell.** The Windows shell is a text-based interface between you or your scripts and the operating system. You communicate with the Windows shell by typing commands in the Windows command prompt; the Windows shell translates these commands into a format that the operating system can process. The operating system then returns any results to the Windows shell, which displays them in the Windows Console.

Accessing the Windows Console in Normal Mode

To access the Windows shell and begin working with it using the command prompt, you must first open a Windows Console. To open a Windows Console on a computer running Windows Vista or Windows 7 (see Figure 1.6), open the Start menu, choose All Programs, choose Accessories, and then choose Command Prompt.