

JavaScript Fundamentals:

Academic Student Guide



**CERTIFIED
INTERNET
WEBMASTER**

EVALUATION COPY

JavaScript Fundamentals

Developer

Brian Danks

Contributor

Lisa Pease

Editor

Tom Graves

Publishers

Scott Evanskey and Joseph A. Servia

Project Managers

Dave De Ponte, Todd Hopkins, and Sheila Ramirez

Trademarks

Prosoft is a trademark of ProsoftTraining. All product names and services identified throughout this book are trademarks or registered trademarks of their respective companies. They are used throughout this book in editorial fashion only. No such use, or the use of any trade name, is intended to convey endorsement or other affiliation with the book. Copyrights of any screen captures in this book are the property of the software's manufacturer.

Disclaimer

ProsoftTraining makes a genuine attempt to ensure the accuracy and quality of the content described herein; however, ProsoftTraining, makes no warranty, express or implied, with respect to the quality, reliability, accuracy, or freedom from error of this document or the products it describes. ProsoftTraining makes no representation or warranty with respect to the contents hereof and specifically disclaims any implied warranties of fitness for any particular purpose. ProsoftTraining disclaims all liability for any direct, indirect, incidental or consequential, special or exemplary damages resulting from the use of the information in this document or from the use of any products described in this document. Mention of any product or organization does not constitute an endorsement by ProsoftTraining of that product or corporation. Data used in examples and exercises is intended to be fictional even if actual data is used or accessed. Any resemblance to, or use of real persons or organizations should be treated as entirely coincidental. ProsoftTraining makes every effort to ensure the accuracy of URLs referenced in all its material, but cannot guarantee that all URLs will be available throughout the life of a course. When this course/disk was published, all URLs were checked for accuracy and completeness. However, due to the ever-changing nature of the Internet, some URLs may no longer be available or may have been re-directed.

Copyright Information

This training manual is copyrighted and all rights are reserved by ProsoftTraining. No part of this publication may be reproduced, transmitted, stored in a retrieval system, modified, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise without written permission of ProsoftTraining, 3001 Bee Caves Road, Austin, TX 78746.

Copyright © 2000 - 2002 by ProsoftTraining
All Rights Reserved

ISBN: 1-58143-646-7



EVALUATION COPY

Table of Contents

Course Description.....	xii
ProsoftTraining Courseware	xiii
Course Objectives	xv
Classroom Setup.....	xvi
System Requirements	xvi
Conventions and Graphics Used in This Book.....	xviii
Lesson 1: Introduction to JavaScript.....	1-1
Pre-Assessment Questions	1-2
Introduction to Scripting	1-3
Origins of JavaScript.....	1-3
JavaScript Characteristics	1-3
JavaScript and Common Programming Concepts	1-6
Java and JavaScript.....	1-7
Server-Side vs. Client-Side Applications	1-10
Annotating Your Code with Comments	1-15
Lesson 1 Review	1-21
Lesson 2: Working with Variables and Data in JavaScript.....	2-1
Pre-Assessment Questions	2-2
Using JavaScript to Communicate with the User	2-4
Using Data More Than Once: Variables.....	2-15
JavaScript Expressions.....	2-21
Operators.....	2-22
Inline Scripting, Simple User Events and the onLoad and onUnload Event Handlers.....	2-29
Keywords and Reserved Words.....	2-30
Lesson 2 Review	2-34
Lesson 3: Functions, Methods and Events in JavaScript.....	3-1
Pre-Assessment Questions	3-2
Functions.....	3-4
Defining a Function	3-4
Calling a Function.....	3-7
User Events and JavaScript Event Handlers	3-21
Methods as Functions	3-22
Lesson 3 Review	3-26
Lesson 4: Controlling Program Flow in JavaScript.....	4-1
Pre-Assessment Questions	4-2
Controlling Decisional Program Flow	4-5
The if...else Statement.....	4-5
The while Statement	4-12
The do...while Statement	4-16
The for Statement.....	4-17
The break Statement	4-21
The continue Statement.....	4-23

The switch Statement	4-26
Lesson 4 Review	4-31
Lesson 5: The JavaScript Object Model	5-1
Pre-Assessment Questions	5-2
The JavaScript Object Model	5-3
Commonly Used Objects	5-5
The window Object	5-5
The document Object	5-21
The with Statement	5-25
The image Object	5-31
The history Object	5-40
The location Object	5-41
The navigator Object	5-43
Lesson 5 Review	5-47
Lesson 6: JavaScript Language Objects	6-1
Pre-Assessment Questions	6-2
Introduction to JavaScript Language Objects	6-4
The String Object	6-4
Additional String Object Methods	6-14
Evaluating Strings	6-14
JavaScript Regular Expressions	6-24
The Array Object	6-28
The Date Object	6-34
Setting and Extracting Time Information	6-41
The Math Object	6-45
Lesson 6 Review	6-50
Lesson 7: Developing Interactive Forms with JavaScript	7-1
Pre-Assessment Questions	7-2
Interactive Forms	7-3
Overview of Form Elements	7-3
Referring to a Form Element	7-5
The form Object	7-5
The button Object	7-7
The checkbox Object	7-9
The text and textarea Objects	7-12
The radio Object	7-19
The select Object	7-23
Multiple-Selection Lists	7-31
Form Validation	7-33
Lesson 7 Review	7-36
Lesson 8: Cookies and JavaScript Security	8-1
Pre-Assessment Questions	8-2
Security and Cookies in JavaScript	8-3
What Are Cookies?	8-3
How Are Cookies Sent?	8-4

Who Can Send Cookies?	8-5
Storing Cookies.....	8-6
Why Use Cookies?.....	8-8
Assigning a Cookie	8-9
Testing for Cookie Presence	8-10
Clearing a Cookie	8-10
Controlling Cookies in the Browser	8-10
JavaScript Security Issues.....	8-23
Lesson 8 Review	8-29
Lesson 9: Controlling Frames with JavaScript.....	9-1
Pre-Assessment Questions	9-2
Using JavaScript with Frames and Windows	9-3
Targeting Frames with JavaScript	9-3
Changing Two or More Frames with JavaScript	9-10
Frames, Functions and Variables.....	9-11
Targeting Windows with JavaScript.....	9-14
Windows, Functions and Variables	9-15
Lesson 9 Review	9-21
Lesson 10: Custom JavaScript Objects.....	10-1
Pre-Assessment Questions	10-2
Creating Custom Objects	10-3
Custom Object Demonstration.....	10-3
Creating a JavaScript Object: The Constructor	10-3
Creating an Instance of a Custom Object	10-5
Creating Object Methods	10-6
Creating Functions for Your Objects.....	10-10
Complex Custom Objects	10-18
Lesson 10 Review	10-20
Appendixes.....	Appendixes-1
Glossary	Glossary-1
Index.....	Index-1
Supplemental CD-ROM Contents.....	Supplemental CD-ROM Contents-1
List of Labs	
Lab 1-1: Creating a JavaScript-enabled page	1-16
Lab 2-1: Using the JavaScript alert() method	2-5
Lab 2-2: Using the JavaScript prompt() method.....	2-8
Lab 2-3: Using the JavaScript confirm() method	2-10
Lab 2-4: Using the JavaScript document.write() method	2-13
Lab 2-5: Storing user data in a JavaScript variable	2-19
Lab 2-6: Assigning and adding variables in JavaScript.....	2-27
Lab 3-1: Creating a user-defined function in JavaScript	3-6
Lab 3-2: Using functions, arguments and return values in JavaScript	3-12

Lab 3-3: Calling a function from within another function in JavaScript.....	3-15
Lab 4-1: Using if statements	4-9
Lab 4-2: Using a while statement	4-13
Lab 4-3: Using a for statement.....	4-19
Lab 4-4: Nesting if and break statements inside a while loop	4-21
Lab 4-5: Using a continue statement	4-25
Lab 5-1: Launching a new window with the open() method	5-12
Lab 5-2: Writing content to new windows	5-15
Lab 5-3: Changing status bar text	5-19
Lab 5-4: Using properties and methods of a remote object	5-26
Lab 5-5: Using the image object	5-36
Lab 6-1: Using String object formatting methods	6-8
Lab 6-2: Applying String methods to text	6-19
Lab 6-3: Creating an Array object	6-32
Lab 6-4: Using the Date object	6-36
Lab 6-5: Creating an onscreen clock	6-42
Lab 7-1: Using a text box, a check box and a button.....	7-16
Lab 7-2: Using radio buttons	7-21
Lab 7-3: Using a select object.....	7-27
Lab 7-4: Using a multiple-selection list.....	7-32
Lab 8-1: Setting, viewing and clearing a cookie with JavaScript using Netscape Navigator	8-16
Lab 8-2: Locking the browser with malicious code.....	8-24
Lab 9-1: Targeting frames	9-8
Lab 9-2: Calling functions from parent and child frames with JavaScript.....	9-12
Lab 9-3: Calling functions from parent and child windows with JavaScript	9-15

List of Figures

Figure 1-1: Comparing Java to JavaScript.....	1-7
Figure 1-2: Lab1-1.htm in Internet Explorer 6.0	1-17
Figure 1-3: Lab1-1.htm in Netscape Navigator 6.2	1-18
Figure 2-1: Alert message.....	2-5
Figure 2-2: Lab2-1.htm displayed following JavaScript statement.....	2-6
Figure 2-3: User prompt dialog box.....	2-8
Figure 2-4: Alert message box.....	2-8
Figure 2-5: Confirm dialog box	2-10
Figure 2-6: Result of confirm() method after clicking OK.....	2-10
Figure 2-7: Result of confirm() method after clicking Cancel	2-11
Figure 2-8: User prompt.....	2-13
Figure 2-9: Lab2-4.htm with customized welcome message.....	2-14
Figure 2-10: Customizing initial prompt	2-14
Figure 2-11: User prompt dialog box.....	2-19
Figure 2-12: Alert message box	2-20
Figure 2-13: Lab2-5.htm with welcome message recalling user's name	2-20
Figure 2-14: Result of JavaScript addition: z=13	2-28
Figure 2-15: Concatenation, not sum.....	2-28

Figure 2-16: Correct sum is indicated.....	2-29
Figure 3-1: Lab3-1.htm.....	3-7
Figure 3-2: Lab3-2.htm.....	3-13
Figure 3-3: Result of function call.....	3-13
Figure 3-4: Result of function call.....	3-14
Figure 3-5: Return value from myFunction().....	3-14
Figure 3-6: Lab3-3.htm.....	3-17
Figure 4-1: Lab4-1.htm.....	4-10
Figure 4-2: Alert dialog box.....	4-10
Figure 4-3: Alert dialog box.....	4-11
Figure 4-4: Alert dialog box.....	4-11
Figure 4-5: Lab4-2.htm.....	4-14
Figure 4-6: Prompt dialog box.....	4-14
Figure 4-7: for loop alert message.....	4-17
Figure 4-8: Output of for loop.....	4-18
Figure 4-9: for loop alert message.....	4-19
Figure 4-10: Output of for loop.....	4-19
Figure 4-11: Lab4-3.htm.....	4-20
Figure 4-12: Lab4-4.htm.....	4-22
Figure 4-13: Prompt dialog box.....	4-22
Figure 4-14: Leap year example.....	4-25
Figure 4-15: Lab4-5.htm.....	4-26
Figure 5-1: JavaScript object hierarchy.....	5-3
Figure 5-2: Lab5-1.htm.....	5-13
Figure 5-3: CIW Web site.....	5-13
Figure 5-4: New window with height and width attributes.....	5-14
Figure 5-5: Small window opened.....	5-16
Figure 5-6: Manipulating status property.....	5-18
Figure 5-7: Lab5-3.htm.....	5-20
Figure 5-8: Lab5-3.htm status bar message.....	5-20
Figure 5-9: Result of using with statement.....	5-26
Figure 5-10: Lab5-4.htm.....	5-28
Figure 5-11: Prompt dialog boxes.....	5-28
Figure 5-12: Alert dialog box.....	5-29
Figure 5-13: New window.....	5-29
Figure 5-14: Alert dialog box in Netscape Navigator.....	5-30
Figure 5-15: Lab5-5.htm.....	5-38
Figure 5-16: Lab5-5.htm with image swap.....	5-38
Figure 6-1: Alert with line breaks in text.....	6-8
Figure 6-2: Lab6-1.htm.....	6-10
Figure 6-3: Links window.....	6-11
Figure 6-4: Output of String prototype demonstration.....	6-13
Figure 6-5: Evaluating strings with string methods.....	6-21
Figure 6-6: Telephone number regular expression.....	6-27
Figure 6-7: questions array has length of 3.....	6-31
Figure 6-8: Lab6-3.htm.....	6-32

Figure 6-9: Lab6-3.htm with sorted array	6-33
Figure 6-10: Alert dialog box	6-33
Figure 6-11: Date object information	6-34
Figure 6-12: Alerts with date information	6-38
Figure 6-13: Date information calculated through script.....	6-38
Figure 6-14: Internet Explorer alert dialog box with year number	6-39
Figure 6-15: Timeclock.htm	6-43
Figure 6-16: Math object alert box sequence.....	6-47
Figure 7-1: Alert with form name/value pairs	7-16
Figure 7-2: Lab7-1.htm	7-17
Figure 7-3: Alert dialog box	7-17
Figure 7-4: Lab7-1.2.htm	7-18
Figure 7-5: Lab7-2.htm	7-22
Figure 7-6: Alert dialog box	7-22
Figure 7-7: Alert dialog box	7-23
Figure 7-8: Lab7-3.htm	7-28
Figure 7-9: Alert dialog box	7-29
Figure 7-10: Lab7-3.1.htm.....	7-30
Figure 7-11: Alert dialog box	7-30
Figure 7-12: Lab7-4.htm	7-32
Figure 7-13: Alert dialog box	7-33
Figure 8-1: Preferences in Netscape Navigator 4.x	8-11
Figure 8-2: Preferences in Netscape Navigator 6.x	8-12
Figure 8-3: Cookie Manager in Netscape Navigator 6	8-13
Figure 8-4: Enabling or disabling cookies in Internet Explorer 4.x	8-14
Figure 8-5: Enabling or disabling cookies in Internet Explorer 5.x	8-15
Figure 8-6: Enabling or disabling cookies in Internet Explorer 6.x	8-16
Figure 8-7: Netscape Navigator cookie warning	8-18
Figure 8-8: Netscape Navigator cookie alert—cookie accepted.....	8-18
Figure 8-9: Netscape Navigator delete cookie confirmation	8-19
Figure 8-10: Netscape Navigator 4.x cookie notice.....	8-19
Figure 8-11: Netscape Navigator 6.x cookie notice.....	8-19
Figure 8-12: Netscape Navigator cookie alert—deleted cookie	8-20
Figure 8-13: Lab8-1.htm	8-20
Figure 8-14: Netscape Navigator alert after canceling deletion of cookie	8-20
Figure 8-15: Lab8-1.htm after canceling deletion	8-21
Figure 8-16: Alert dialog box	8-25
Figure 8-17: Close Program menu.....	8-25
Figure 8-18: End Task message.....	8-26
Figure 9-1: 3frames-1.htm	9-5
Figure 9-2: Major.htm.....	9-8
Figure 9-3: New banner loaded into banner frame	9-9
Figure 9-4: New page loaded into main frame	9-9
Figure 9-5: CIW Certified Web site.....	9-10
Figure 9-6: Major.htm.....	9-13
Figure 9-7: Alert dialog box with value of myParentVar	9-13

Figure 9-8: Alert dialog box defined in myParentFunction()	9-13
Figure 9-9: Alert dialog box defined as return value of myParentFunction()	9-14
Figure 9-10: Lab9-3.htm	9-17
Figure 9-11: Child window	9-17
Figure 9-12: Parent function called from child window	9-18
Figure 9-13: Child window function called	9-18
Figure 10-1: Product list page	10-6
Figure 10-2: After clicking Get Info	10-10
Figure 10-3: Search item for ID	10-12
Figure 10-4: Search result for invalid entry	10-12
Figure 10-5: Result of clicking Show All Products button	10-14
Figure 10-6: Result of selecting CIW Polo Shirt link	10-14

List of Tables

Table 1-1: Comparison of JavaScript and Java	1-8
Table 1-2: JavaScript versions and browser support	1-9
Table 2-1: JavaScript data types	2-15
Table 2-2: JavaScript literal types	2-16
Table 2-3: JavaScript operators	2-22
Table 3-1: JavaScript user events	3-21
Table 3-2: JavaScript event handlers	3-22
Table 5-1: Properties, methods and event handlers of window	5-5
Table 5-2: window attributes accessible in open() method	5-10
Table 5-3: Properties and methods of document	5-21
Table 5-4: Comparison of code using with statement	5-26
Table 5-5: Properties and event handlers of image object	5-34
Table 5-6: Properties and methods of history	5-40
Table 5-7: Properties of location	5-42
Table 5-8: Properties and methods of navigator	5-43
Table 6-1: String object formatting methods	6-6
Table 6-2: Special characters in JavaScript	6-7
Table 6-3: Methods of Date object	6-35
Table 6-4: Methods of Math object	6-45
Table 6-5: Properties of Math object	6-46
Table 7-1: Form elements	7-3
Table 7-2: Features of form object	7-6
Table 7-3: Features of button object	7-7
Table 7-4: Features of checkbox object	7-11
Table 7-5: Features of text and textarea objects	7-13
Table 7-6: Features of radio object	7-20
Table 7-7: Properties of select object	7-25

Course Description

Welcome to *JavaScript Fundamentals*, a two-day course designed to teach you the features of the JavaScript language. This course will empower you with the skills to design client-side, platform-independent solutions that greatly increase the value of your Web site. The first day of this course teaches you foundational JavaScript skills. The second day will build on your knowledge and present solutions for more functional and exciting Web pages. You will learn how to communicate with users, script for the JavaScript object model, control program flow, validate forms, animate images, target frames, and create cookies. By the end of this course, you will understand and use the most popular applications of JavaScript.

Length

JavaScript Fundamentals is a 12-hour course that can be implemented in two days.

Series

JavaScript Fundamentals is the first course in the CIW Web Languages series. CIW Web Languages consists of the following 2 courses:

- *JavaScript Fundamentals*
- Perl Fundamentals

Prerequisites

Students must have completed the *CIW Foundations Series* or be able to demonstrate equivalent Internet knowledge.

ProsoftTraining Courseware

This coursebook was developed for instructor-led training and will assist you during class. Along with comprehensive instructional text and objectives checklists, this coursebook provides easy-to-follow hands-on labs and a glossary of course-specific terms. It also provides Internet addresses needed to complete some labs, although due to the constantly changing nature of the Internet, some addresses may no longer be valid.

The student coursebook is organized in the following manner:

course title	
table of contents	
list of labs	
list of figures	
list of tables	
lessons	
lesson objectives	
pre-assessment questions	
narrative text	
<input checked="" type="checkbox"/> graphics	
<input checked="" type="checkbox"/> tables and figures	
<input checked="" type="checkbox"/> warnings	
<input checked="" type="checkbox"/> tech notes	
labs	
<input checked="" type="checkbox"/> graphics	
<input checked="" type="checkbox"/> tables and figures	
<input checked="" type="checkbox"/> warnings	
<input checked="" type="checkbox"/> tech notes	
lesson summary	
lesson review	
appendixes	
glossary	
index	
supplemental CD	

When you return to your home or office, you will find this coursebook to be a valuable resource for applying the skills you have learned. Each lesson concludes with questions that review the material. Lesson review questions are provided as a study resource only and in no way guarantee a passing score on CIW exams.

The course is available in either an academic or a learning center version, and each version has an instructor book and a student book. Check your book to verify that you have the correct version, and whether it is an instructor or a student book. Following is a brief description of each version.

- **Academic:** Designed for students in an academic classroom environment; typically taught over a quarter (10-week) or semester (16-week) time period. Example syllabi for both timeframes are included on the instructor CD-ROM. The instructor's book and CD-ROM contain all answers, as well as activities (pen-and-paper-based labs), optional labs (computer-based labs), quizzes, a course assessment, and handouts for the instructor to assign during class or as homework. No answers exist in the student book or on the student CD-ROM. Students must obtain answers from the instructor.
- **Learning Center:** Designed for students in a learning center classroom environment; typically taught over a one- to five-day time period (depending on the length of the course). An example implementation table is included on the instructor CD-ROM. Similar to the academic version, the instructor's book and CD-ROM contain all answers, as well as activities (pen-and-paper-based labs), optional labs (computer-based labs), quizzes, a course assessment, and handouts for the instructor to assign during class or as homework. However, the student CD-ROM also contains answers, including those to the pre-assessment questions, labs, review questions, activities, optional labs, quizzes, and the course assessment.

Course Objectives

After completing this class, you will be able to:

- Describe the origins of JavaScript and list its key characteristics.
- Communicate with users using JavaScript.
- Define and call JavaScript functions.
- Control program flow.
- Explain and use the JavaScript object model.
- Identify and use the JavaScript language objects.
- Use JavaScript with HTML form controls.
- Define and utilize cookies.
- Discuss security issues relevant to JavaScript.
- Create custom JavaScript objects.

Classroom Setup

Your instructor has probably set up the classroom computers based on the system requirements listed below. Most software configurations on your computer are identical to those on your instructor's computer. However, your instructor may use additional software to demonstrate network interaction or related technologies.

System Requirements

Hardware

The following table summarizes the hardware requirements for all courses in the CIW program. Each classroom should be equipped with enough personal computers to accommodate each student and the instructor with his or her own system.

Note: The CIW hardware requirements are similar to the lowest system requirements for Microsoft implementation (Level 1 requirements) except that CIW requires increased hard disk space (8 GB) and RAM (128 MB). This comparison may be helpful for the many training centers that implement CIW and are also CTEC because personnel at these centers are familiar with the Microsoft hardware specifications.

CIW hardware specifications	Greater than or equal to the following
Processor	Intel Pentium II (or equivalent) personal computer with processor speed greater than or equal to 300 MHz
L2 cache	256 KB
Hard disk	8-GB hard drive
RAM	At least 128 MB
CD-ROM	32X
Network interface card (NIC)	10BaseT or 100BaseTX (10 or 100 Mbps)
Sound card/speakers	Required for instructor's station, optional for student stations
Video adapter	At least 4 MB
Monitor	15-inch monitor
Network hubs	Two 10-port 10BaseT or 100BaseTX (10 or 100 Mbps) hubs
Router	Multi-homed system with three NICs (Windows NT 4.0/2000 server)*

* Must meet universal CIW hardware requirements.

Software

The recommended software configurations for computers used to complete the labs in this book are as follows.

To be installed before class:

- Microsoft Windows Millennium Edition (Me) or Windows 2000.
- Internet Explorer 4.0 (or later) or Netscape Navigator 4.0 (or later).
- A standard text editor (usually available with the operating system).

Connectivity

Internet connectivity is required for this course. The minimum requirement is a modem and an Internet service Provider (ISP) account. However, you will experience improved results with a 56-Kbps modem with a Point-to-Point Protocol (PPP) account through an ISP, or a direct Internet connection.

Conventions and Graphics Used in This Book

The following conventions are used in Prosoft coursebooks.

Terms	Technology terms defined in the margins are indicated in bold the first time they appear in the text. Not every word in bold is a term requiring definition.
Lab Text	Text that you enter in a lab appears in bold . Names of components that you access or change in a lab also appear in bold .
Notations	<i>Notations or comments regarding screenshots, labs or other text are indicated in italic type.</i>
Program Code or Commands	Text used in program code or operating system commands appears in the Lucida Sans Typewriter font.

The following graphics are used in Prosoft coursebooks.



Tech Notes point out exceptions or special circumstances that you may find when working with a particular procedure. Tech Notes that occur within a lab are displayed without the graphic.



Tech Tips offer special-interest information about the current subject.



Warnings alert you about cautions to observe or actions to avoid.



This graphic signals the start of a lab or other hands-on activity.



This graphic indicates a line of code that is completed on the following line.

Lesson 1:

Introduction to JavaScript

Objectives

By the end of this lesson, you will be able to:

- ✧ Describe the origins of JavaScript.
- ✧ List the key JavaScript characteristics.
- ✧ Describe the differences between Java and JavaScript.
- ✧ Discern among JavaScript, JScript and VBScript.
- ✧ Differentiate among server-side and client-side JavaScript applications.
- ✧ Embed JavaScript into HTML.
- ✧ Use the JavaScript comment tags.

Pre-Assessment Questions

1. Which of the following describes a platform intended to run client-side JavaScript?
 - a. Microsoft Internet Explorer 3.0 or later
 - b. Netscape Navigator 1.0 or later
 - c. Netscape Navigator 2.0 or later
 - d. Microsoft Internet Explorer 2.0 or later
2. Which of the following properly references an external JavaScript file?
 - a. `<SCRIPT TYPE="text/javascript" script_src="myJSCode.js">`
 - b. `<SCRIPT TYPE="text/javascript" source="myJSCode.js">`
 - c. `<SCRIPT TYPE="text/javascript" src="myJSCode.js">`
 - d. `<SCRIPT TYPE="text/javascript" script_source="myJSCode.js">`
3. Briefly describe the difference between an event-driven programming model and a procedural programming model.

Introduction to Scripting

When the World Wide Web first became popular, Hypertext Markup Language (HTML) was the only language an author could use to create Web pages. HTML is not a programming language, but a "markup" language and has many limitations. HTML positions text and graphics on a Web page but offers limited interactivity within the Web page. Most computer users, whether they use Windows, Macintosh, UNIX or some combination thereof, are now accustomed to graphical application interfaces. They click buttons to execute command sequences, enter values into text boxes, and choose from menu lists. This increase in user abilities and expectations has resulted in a continual improvement of HTML, as well as the advent of powerful scripting languages such as JavaScript.

Origins of JavaScript

Netscape Corporation developed the JavaScript language. JavaScript is not a stand-alone programming language like Java. When JavaScript was first developed, its name was LiveScript, but Netscape contracted with Sun Microsystems to name it JavaScript, more due to Java's popularity than any similarity between the two languages.

To run properly, client-side JavaScript is written within HTML documents. Server-side JavaScript, called LiveWire, can work with back-end server processes. Whether used for client-side or server-side solutions, JavaScript allows programmers to add interactivity to Web pages without using server-based applications, such as Common Gateway Interface (CGI) programs.

JavaScript was first supported in Navigator version 2.0, and has since gained universal support in such browsers as Internet Explorer 3.0 and later, Mosaic 2.0 and later, and Lotus Personal Web Client 3.0 and later.

JavaScript Characteristics

Before you start writing in JavaScript, you should take a close look at the features of this language.

Perl

A script programming language commonly used for Web server tasks and CGI programs.

Tool Command Language (Tcl)

An interpreted script language used to develop applications such as GUIs, prototypes and CGI scripts.

REXX

A procedural programming language used to create programs and algorithms.

JavaScript is a scripting language

A scripting language is a simple programming language designed to enable computer users to write useful programs easily. Scripting languages, including **Perl**, **Tcl** and **REXX**, are interpreted, meaning that they are not compiled to any particular machine or operating system. This feature makes them platform-independent.

One of the key uses for scripting languages such as JavaScript is to allow more complex programs, created by programming languages such as C and C++, to work together. JavaScript is one of the more popular languages, and is uniquely suited for this purpose.

If you have ever written a macro in Microsoft Excel or used WordBasic to perform some task in a Microsoft Word document, you have already used a scripting language. Smaller and less powerful than full programming languages, scripting languages provide easy functionality. JavaScript syntax is similar to that of C or Pascal.

JavaScript is object-based, not object-oriented

Object-oriented is a common term in programming languages. An object-oriented program is a collection of individual objects that perform different functions, rather than a sequence of statements that collectively perform a specific task. These objects are usually related in a hierarchical manner, in which new objects and subclasses of objects inherit the properties and methods of the objects above them in the hierarchy. JavaScript is not object-oriented because it does not allow for object inheritance and subclassing in the traditional sense. However, JavaScript is an *object-based* language because it derives functionality from a collection of built-in objects. With JavaScript, you can also create your own objects. The concept of objects will be discussed further later in this lesson.

JavaScript is event-driven

The World Wide Web is based upon an event-driven model. For example, whenever you click an item on a Web page, an event occurs. The previous programming model was the procedural model, in which the user (if there is one) is expected to interact with the program in a fairly sequential manner. On a Web page, however, the user is in control and can click or not click, move the mouse or not move the mouse, or change the URL at will. Because of the unpredictability of a user's actions, programming modules (called subroutines or functions) can be created that are independent of each other and do not require any sequential set of operations.

Events can trigger functions. Event triggers can be as simple as the user clicking a button, clicking or moving the mouse over a hyperlink, or entering text into a text field. Scripting can be tied to any of these events. You will learn to use JavaScript to instruct the browser what to do or display using the event that you designate as the trigger for the script.

Client-side JavaScript is part of the text within your HTML document. When your browser retrieves a scripted page, it executes the JavaScript programs and performs the appropriate operations in response to user events.

JavaScript is platform-independent

user agent

The W3C term for any application, such as a Web browser or help engine, that renders HTML for display to users.

Because JavaScript programs are designed to run within HTML documents, they are not tied to any specific hardware platform or operating system. However, JavaScript programs are tied to a specific **user agent**. Generally, these user agents are browsers. Theoretically, you can implement the same JavaScript program on any current user agent, such as Netscape Navigator 2.0 or later, or Internet Explorer 3.0 or later.

Keep in mind that each user agent tends to implement JavaScript differently. Different vendors and user agent versions can complicate your JavaScript implementation. Because you usually cannot guarantee that a user will access your JavaScript code using a specific user agent, take care to create code that will run on as many platforms as possible. JavaScript provides a way for you to determine the user agent used to access your programs. This concept will be explored later in this course.

JavaScript enables quick development

graphical user interface (GUI)

A program that provides graphical navigation with menus and screen icons.

Because JavaScript does not require time-consuming compilation, scripts can be developed quickly. This advantage is enhanced by the fact that most of the interface features, such as forms, frames and other **graphical user interface (GUI)** elements, are handled by the browser and HTML code. JavaScript programmers need not worry about creating or handling these elements of their applications.

JavaScript is relatively easy to learn

JavaScript does not have the complex syntax and rules associated with Java. Even if you do not know any other programming language, learning JavaScript will not be difficult.

JavaScript and Common Programming Concepts

Some key JavaScript concepts may not make sense right away. However, as this course examines various examples and labs, you will acquire a better understanding of these concepts.

Scripting languages are subsets of larger, more complicated languages. They provide less functionality than full programming languages, but are usually easier to learn. Your investment in learning a scripting language is valuable if you decide to learn a full programming language because you will gain basic conceptual awareness of common programming practices.

Objects, properties and methods

object

A programming function that models the characteristics of abstract or real "objects" using classes.

property

A descriptive characteristic of an object, such as color, width or height, that the programmer stipulates in the creation of the object.

value

The specific quality such as color, width or height that belongs to the property of an object.

method

An action that can be performed by an object.

Serious programmers tend to write code in C++, Visual Basic or Java. These three high-level languages provide rich functionality to the program developer. They are also "object-oriented" languages. In programming, **objects** encapsulate predesignated attributes and behavior. They are often grouped with similar objects into classes.

Like real-life objects, JavaScript objects have certain attributes and behaviors. Developers refer to attributes and behaviors with three other terms: **properties**, **values**, and **methods**. Properties represent various attributes of an object, such as height, color, font size, sentence length and so forth. Values represent the specific qualities of properties. For instance, the statement `color="red"` assigns a value to a property. Methods are the actions that an object can be made to perform, such as a calculation, an onscreen move or the writing of text in a window. Methods often describe the actions that an object performs with its properties.

To further explain objects, properties, values and methods, consider a pen in terms of object-based programming. It is clearly an object. A pen has definite, discernable properties, such as a length, ink color, point style and so forth. Two pens may have similar properties, yet they may have different values for those properties: all pens will have a color (i.e., property), but all pens will not have the same color (i.e., value).

A pen has methods as well. It can write, spin, flip, and have its cap removed or replaced. A developer creating a virtual pen object would simulate natural attributes and behavior so that, for example, if users try to write with pens, they might receive errors directing them to first remove the caps.

A sentence on a page is another example of an object. You can think of a sentence as a string of words. But to a programmer, that sentence might be an object that has minimum properties of characters, color, and font. The values would include number of characters (or sentence length), color type and font style. In addition, you may want to invoke a method in which you can convert that sentence to all uppercase or all lowercase letters.

Ultimately, objects can be defined as collections of properties, with collections of methods that operate on those properties. As you work and develop an understanding of JavaScript, you will see many examples of objects along with their properties and methods.

Java and JavaScript

Although the names are similar, Java and JavaScript are different languages. Java is a full-fledged object-oriented programming language. Developed by Sun Microsystems, Java can be used to create stand-alone applications and a special type of mini-application called a Java applet. Applets are written in Java, compiled, and then referenced via the <APPLET> tag (or <OBJECT> in HTML 4.0) in a Web page. Applets can provide a great variety of added functionality to Web sites.

JavaScript is an object-based scripting language. Although it uses some of Java's expression syntax and basic program flow controls, JavaScript stands alone and does not require Java. In fact, any similarities between the two are due to their use of objects. JavaScript was not developed by the same company, nor was it developed with Java in mind. Figure 1-1 illustrates how the similarities are tangential.

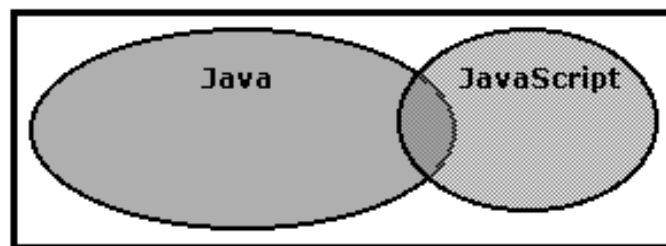


Figure 1-1: Comparing Java to JavaScript

Table 1-1 further compares JavaScript and Java.

Table 1-1: Comparison of JavaScript and Java

JavaScript	Java
Interpreted (not compiled) by client.	Compiled on server before execution on client.
Object-based; code uses built-in, extensible objects, but no classes or inheritance.	Object-oriented; applications and applets consist of object classes with inheritance.
Variable data types not declared (loose typing).	Variable types must be declared (strong typing).
Dynamic binding; object references checked at run time.	Static binding; object references must exist at compile time.
Secure; cannot write to hard disk.	Secure; cannot write to hard disk.
Code integrated with and embedded in HTML.	

JavaScript and VBScript

JavaScript and VBScript are scripting languages that have similar purposes. Both extend the capabilities of static Web pages.

JavaScript was the first scripting language developed for Web page design. It has been incorporated into the popular Netscape Navigator browsers since version 2.0. VBScript is the Microsoft Web-scripting language, based on the powerful and popular Visual Basic language.

JavaScript is a strong object-based language that relies, for much of its functionality, on objects and their attendant methods and properties.

VBScript relies much less upon the traditional object classes and much more on dynamic, built-in customizing functions. Is one scripting language better? It depends entirely on the situation and the programmer's knowledge of that particular language. Programmers must consider that client-side VBScript is only supported by the Microsoft Internet Explorer browser.

Visual Basic and VBScript

Just as JavaScript is a separate language that has similarities to Java, VBScript is a language that has similarities to Visual Basic. In contrast, however, VBScript is a subset of the Visual Basic language.

JavaScript, JScript and ECMAScript

The Microsoft implementation of the Netscape JavaScript language is called JScript. Several differences exist between these two implementations. Although some of the differences are minor, others cause compatibility problems.

Netscape, the inventor of JavaScript, announced in 1997 that the European Computer Manufacturer's Association (ECMA) had approved JavaScript as an international standard. In some circles, JavaScript is now officially known as ECMAScript, and many hope that it will become widely accepted. The ECMA standard intends to diminish the differences between JavaScript and JScript. The following description is quoted from the ECMA Web site:

"This ECMA Standard is based on several originating technologies, the most well known being JavaScript (Netscape) and JScript (Microsoft)."

For more information about ECMAScript, consult the ECMA home page at <http://www.ecma.ch/>. This site refers to the ECMAScript standard as Standard ECMA-262. The most recent specification is the third edition, published in December 1999.

JavaScript versions

Currently, six versions of JavaScript exist. Table 1-2 lists the various versions of JavaScript and the implementations of Netscape Navigator and Microsoft Internet Explorer that support each version. Corresponding versions of JScript and support environments are also listed.

Table 1-2: JavaScript versions and browser support

Version	Netscape Navigator Support	Microsoft Internet Explorer Support
JavaScript 1.0 JScript 1.0	Navigator 2.x	Internet Explorer 3.x
JScript 2.0	N/A	Microsoft Internet Information Server 1.0
JavaScript 1.1	Navigator 3.x	N/A

Table 1-2: JavaScript versions and browser support (cont'd)

Version	Netscape Navigator Support	Microsoft Internet Explorer Support
JavaScript 1.2 JScript 3.0	Navigator 4.0 through 4.05	Internet Explorer 4.x Microsoft Internet Information Server 4.0 Microsoft Windows Scripting Host 1.0
JavaScript 1.3	Navigator 4.06 through 4.7x	N/A
JScript 4.0	N/A	Microsoft Visual Studio 6.0
JavaScript 1.4 JScript 5.0	Did not map to a release of Navigator	Internet Explorer 5.x
JavaScript 1.5	Navigator 6.0	N/A
JScript 5.5	N/A	Microsoft Internet Information Services 5.0

One of the challenges of creating JavaScript programs is remembering the various features offered by the different versions of the language. You must also be aware of your target audience and the browsers they most likely use. Currently, Netscape Navigator and Microsoft Internet Explorer are the two most popular browsers. Most Web applications are built to target these two browsers.

Fortunately, the core functionality offered by JavaScript does not change from version to version. As new versions of the language are released, new features are added, but the basic features remain the same.

Server-Side vs. Client-Side Applications

The focus of this course is client-side JavaScript. You can use client-side JavaScript, for example, to validate specific form fields of data to make sure the client has entered the required fields. By using JavaScript wisely, you can begin processing client information before it reaches the more resource-intensive server processes.

JavaScript can also be used in server-side solutions. For instance, after the fields of data are submitted to a server, server-side JavaScript can be used to parse and disseminate the information.

The key to using JavaScript properly is remembering that the wisest uses of JavaScript unify client-side and server-side solutions.

Server-side applications and LiveWire

LiveWire, JavaScript's server-side solution, enables you to connect Web pages to databases, as well as enables server-side image maps and saves client state so the computer will remember where the client is during a multi-page process. LiveWire is an add-on package that works with Netscape Enterprise Server.

LiveWire technology includes server-side programming objects that allow the programmer to detect and respond to information sent from client browsers. These objects also enable the programmer to work with and manipulate relational databases such as Informix, Oracle, and Sybase. LiveWire also provides a WYSIWYG (what you see is what you get) editor/browser and a graphical Web site manager. JavaScript can be used on either the server side or the client side to work with LiveWire. However, server-side JavaScript will function only if LiveWire is installed.

Currently, JavaScript does not support direct database access without LiveWire.

Client-side applications

This course teaches you how to implement client-side applications. Client-side JavaScript can be used to build many scalable solutions that allow you to send messages to users, launch new windows, work with frames, create your own objects and more.

Embedding JavaScript into HTML

JavaScript resides within an HTML document. Authors usually place it into an HTML document using the <SCRIPT> tag. You can add script to the head or the body section (or both) of an HTML document. Later, you will learn how to embed scripting instructions directly into certain HTML tags as well, a technique called inline scripting.

The basic structure of an HTML file with JavaScript is as follows (JavaScript indicated in bold):

```
<HTML>
<HEAD>
<TITLE>Page Title</TITLE>

<SCRIPT LANGUAGE="JavaScript">
<!--

//JavaScript code goes here

// -->
</SCRIPT>

</HEAD>
<BODY>
HTML page text

<SCRIPT LANGUAGE="JavaScript">
<!--

//JavaScript goes here too

// -->
</SCRIPT>

HTML page text
</BODY>
</HTML>
```



Notice that the HTML comment tag is used to "comment out" or hide the text of the script. You use this in case the person browsing your page has an old browser that cannot execute JavaScript code. The comment tags prevent the specified code from displaying in the browser window. As more users upgrade their browsers, the comment tag will become less necessary for this purpose.

Notice the special addition in the ending comment tag line. The // characters at the beginning of the line are how JavaScript comments out text. JavaScript does not know how to handle the ending comment tag because it consists of two JavaScript operators; therefore you must hide that line from the JavaScript code. Failure to include those lines results in an error message when the page is loaded in some browsers, and may keep the script from executing properly. The JavaScript operators mentioned will be discussed in a later lesson.

The HTML 4.0 TYPE attribute

The HTML 4.0 specification deprecated the LANGUAGE attribute for the <SCRIPT> tag. In its place, the TYPE attribute is recommended. This attribute is used as follows:

```
<SCRIPT TYPE="text/javascript">
```

The value for TYPE is in the form of a MIME type descriptor. Note that only browsers such as Netscape Navigator 6.0 and higher, and Microsoft Internet Explorer 5.0 and higher, support the TYPE attribute. The LANGUAGE attribute can be used as needed because it is still supported by all JavaScript-enabled browsers.

Denoting JavaScript versions

As noted earlier, several different versions of JavaScript exist. If you need to ensure that browsers know which version of JavaScript you are using, this information can be placed in the <SCRIPT> tag using the LANGUAGE attribute. The following example demonstrates this syntax:

```
<HTML>
<HEAD>
<TITLE>Page Title</TITLE>

<SCRIPT LANGUAGE="JavaScript1.5">
<!--

//JavaScript 1.5 code goes here

// -->
</SCRIPT>

</HEAD>
<BODY>
HTML page text
</BODY>
</HTML>
```

In this example, the developer forces the user agent to use the 1.5 version of JavaScript. If the user agent cannot execute JavaScript 1.5 code, the entire script block is ignored.



If the LANGUAGE attribute is omitted from the <SCRIPT> tag, Netscape Navigator and Microsoft Internet Explorer will typically assume the most current version of JavaScript.

External scripts

You can also include script as an external file. This strategy is helpful if your code is more complex, if you plan on revising the code often, or if you plan on using the same code in multiple pages.

To create an external JavaScript file, use the .js file name extension. The .js file consists of native JavaScript code without the HTML <SCRIPT> tag. You can then include code similar to the following within the <HEAD> or <BODY> tags (note the use of the SRC attribute):

```
<SCRIPT LANGUAGE="JavaScript" SRC="JavaScriptCode.js">
<!--

/*
Avoid adding embedded JavaScript code. It will
not be executed if the .js file is unavailable.
*/

// -->
</SCRIPT>
```

The browser will automatically read the code written in the .js file as if it were placed between the <SCRIPT> tags. As noted in the code, any additional embedded JavaScript code will not be executed if the .js file is not available in Netscape Navigator 4.x and higher and in Microsoft Internet Explorer 4.x and higher. If any additional code needs to be executed in the file, that code should reside in a different <SCRIPT> block. The JavaScript comment tags used in the previous examples will be discussed in detail in the next section.



Note that Netscape Navigator 2.x and Microsoft Internet Explorer 3.x do not support external JavaScript files. Use caution if your target audience uses these browsers. Also note that the Web server hosting the application must be aware of the .js extension to associate the proper file with the HTML page.

The <NOSCRIPT> tag

You should be aware that your users may have older browsers that do not support JavaScript, or may disable execution of JavaScript code within their browsers. If so, you can use the HTML <NOSCRIPT> tag to render content for these users. An example of the <NOSCRIPT> tag syntax follows:

```
<SCRIPT LANGUAGE="JavaScript">
<!--

//JavaScript to insert information

//-->
</SCRIPT>
```



```
<NOSCRIPT>
  Click here for
  <A HREF="http://www.yourCompany.com/info/info.htm">info.</A>
</NOSCRIPT>
```

Annotating Your Code with Comments

As your scripts become longer and more complex, they will become harder to read and debug. For this reason, successful programmers place comments in their code to help remind them of the purpose and function for each major section of code.

You know that HTML allows you to use comment tags to place comments in the document that have no effect on the document's appearance to the user. Similarly, you can also place comments in your JavaScript code.

JavaScript uses two types of comment indicators. One indicator delineates a comment on a single line of script (`//`). The other type of comment indicator is used for multiple-line comments (`/* ... */`). The following sections provide further discussion of each.

Single-line comment indicator

You have already used the `//` to comment out the ending HTML comment tag. However, the single-line comment indicator is not limited to just this use. You can use this method to add comments to a whole or partial line as demonstrated in the following code excerpt:

```
<SCRIPT LANGUAGE="JavaScript">
<!--

  // Variables defined here

  var firstNum = 20
  var secondNum = 0   // this value will change

  //-->
</SCRIPT>
```

In this example, the portion of the coding from the `//` to the end of the line will be ignored by the JavaScript interpreter.

Multiple-line comment indicator

Eventually, you will need to use a comment that extends beyond a single line. To perform this task, enclose the area that will not execute with the `/*` and `*/` indicators. Note that the syntax for this comment is exact.

The following is an example of a multi-line comment:

```
<SCRIPT LANGUAGE="JavaScript">
<!--

/*
  The function addNumbers( ) is used to calculate the
  two numbers that are supplied to the function by the
  user.
*/

function addNumbers() {
  //code for function
}

//-->
</SCRIPT>
```



You can also use comments to prevent a section of code from executing if you need to troubleshoot your script. If you enclose the section of suspect script within comment indicators, JavaScript will ignore that section when your script executes.

The following lab will provide an opportunity to create a JavaScript-enabled HTML page. Any text editor can be used to create this page. Check with your instructor for special instructions concerning the editor to be used for this course.

Before beginning this lab, copy the Student_Files folder from the supplemental disk to your desktop.



Lab 1-1: Creating a JavaScript-enabled page

In this lab, you will create your first JavaScript page, which will introduce two JavaScript objects using a method of one and two properties of the other. The first object is the document object and will use its `write` method. The second object is the navigator object and will use its `appName` and `appVersion` properties. Both objects will be fully discussed in a later lesson.

1. **Editor:** Open the **lab1-1.htm** file from the Lesson 1 folder of the Student_Files directory. Enter the code indicated in bold:

```
<HTML>
<HEAD>
<TITLE>Lab 1-1</TITLE>
</HEAD>

<BODY>

<!-- Create <SCRIPT> block here -->

<SCRIPT LANGUAGE="JavaScript">
<!--

document.write(navigator.appName);
document.write("<P>");
document.write(navigator.appVersion);

//-->
</SCRIPT>

</BODY>
</HTML>
```

2. **Editor:** Save **lab1-1.htm**.
3. **Browser:** Browse to **lab1-1.htm**. If you are not sure how to access the file, ask your instructor for help. Your screen should resemble Figure 1-2, depending on the browser you are using for this course. Figure 1-2 shows lab1-1.htm in Microsoft Internet Explorer 6.0.

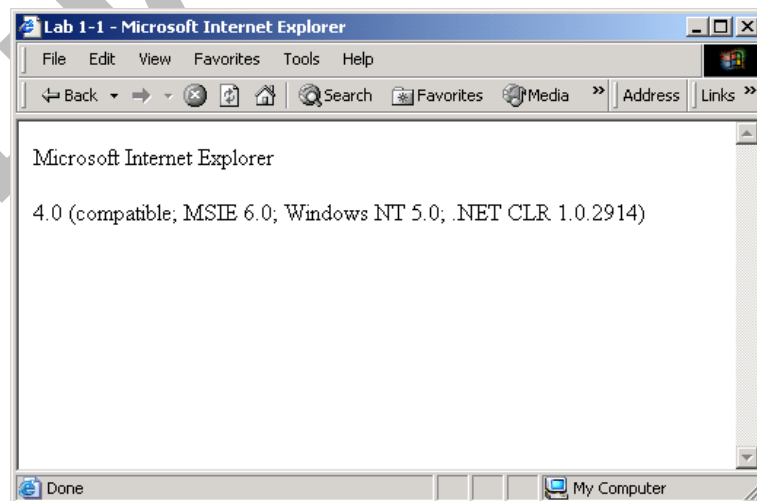


Figure 1-2: Lab1-1.htm in Internet Explorer 6.0

4. Browser: Figure 1-3 shows lab1-1.htm in Netscape Navigator 6.2.

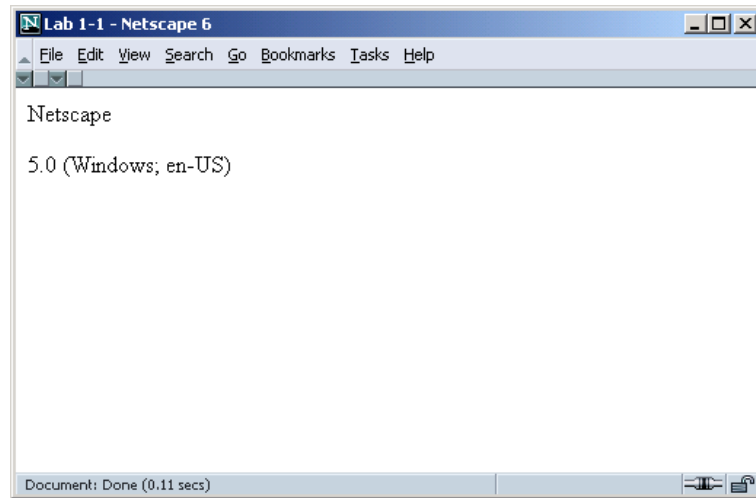


Figure 1-3: Lab1-1.htm in Netscape Navigator 6.2

As you can see, differences exist in the format that each browser uses for the output of the JavaScript statements. This is one example of the differences in implementation of JavaScript from browser to browser.

In this lab you used a `document.write()` statement. The `document` object's `write()` method is used to output data to the HTML stream. You also used the `navigator` object's `appName` and `appVersion` properties. The `appName` property returns a string value indicating the name of the client browser. The `appVersion` property returns a string value indicating the version number of the client browser, as well as the client's platform. These objects, method and properties will be discussed later in this course.

Notice that in the `document.write()` statements, the text `navigator.appName` and `navigator.appVersion` were not inside quotation marks, whereas the HTML `<P>` tag was inside quotation marks. The two lines of code using the `navigator` object are evaluated text. In other words, the JavaScript interpreter dynamically supplies the appropriate text when the script is executed. Therefore, that text was not inside quotation marks. The literal `<P>` tag is static text. Its value is known before the script runs, so it is placed inside quotation marks. These concepts will be further demonstrated and discussed later in this course.

Lesson Summary



Application project

The World Wide Web provides many valuable JavaScript resources. Use any good search engine and locate several JavaScript resources. Also, resources are listed in an appendix. Begin investigating sites that provide language documentation, tutorials and code examples. Become familiar with the various resources on the World Wide Web. Determine and document which sites provides the most information, which is the easiest to use, and which will benefit you most as you learn the JavaScript language.



Skills review

Computer users' expectations and the demand for sophisticated graphical application interfaces have resulted in the advent of powerful scripting languages, such as JavaScript. JavaScript was first supported in Navigator version 2.0 and has since gained wide support in most popular browsers. Like other scripting languages, JavaScript is interpreted, not compiled to any particular machine or operating system. Scripting languages are subsets of larger, more complicated languages. JavaScript is an object-based language because it derives functionality from a collection of built-in objects. JavaScript is event-driven and platform-independent. In programming, objects encapsulate predesignated attributes and behavior. Developers refer to attributes and behaviors by three specific terms: properties, values and methods. Properties represent various attributes of an object. Values represent the specific qualities of properties. Methods are the actions that an object can be made to perform. Although the names are similar, Java and JavaScript are different languages. Java is a full-fledged object-oriented programming language developed by Sun Microsystems. JavaScript is an object-based scripting language that stands alone and does not require Java. Currently, six versions of JavaScript exist. JavaScript can be used for client-side and server-side solutions. JavaScript is usually placed within an HTML document using the `<SCRIPT>` tags. You can also include JavaScript as an external file. JavaScript uses two types of comment indicators, one for single line comments (`//`) and one for multiple-line comments (`/* . . . */`).

Now that you have completed this lesson, you should be able to:

- ✓ Describe the origins of JavaScript.
 - ✓ List the key JavaScript characteristics.
 - ✓ Describe the differences between Java and JavaScript.
 - ✓ Discern among JavaScript, JScript and VBScript.
 - ✓ Differentiate between server-side and client-side JavaScript applications.
 - ✓ Embed JavaScript into HTML.
 - ✓ Use the JavaScript comment tags.
-

EVALUATION COPY

Lesson 1 Review

1. What advantage do scripting languages such as JavaScript offer over HTML?

2. What advantage do scripting languages such as JavaScript offer over programming languages such as Java?

3. Name at least three key characteristics of JavaScript.

4. Where does client-side JavaScript code reside, and how is it activated?

5. In programming, what is an object?

6. In programming, developers refer to attributes and behaviors by what other three terms?

7. How do JavaScript, VBScript and JScript differ?

8. What is the name of JavaScript's server-side technology?

9. What two types of comment indicators can you use to annotate your JavaScript code?

EVALUATION COPY