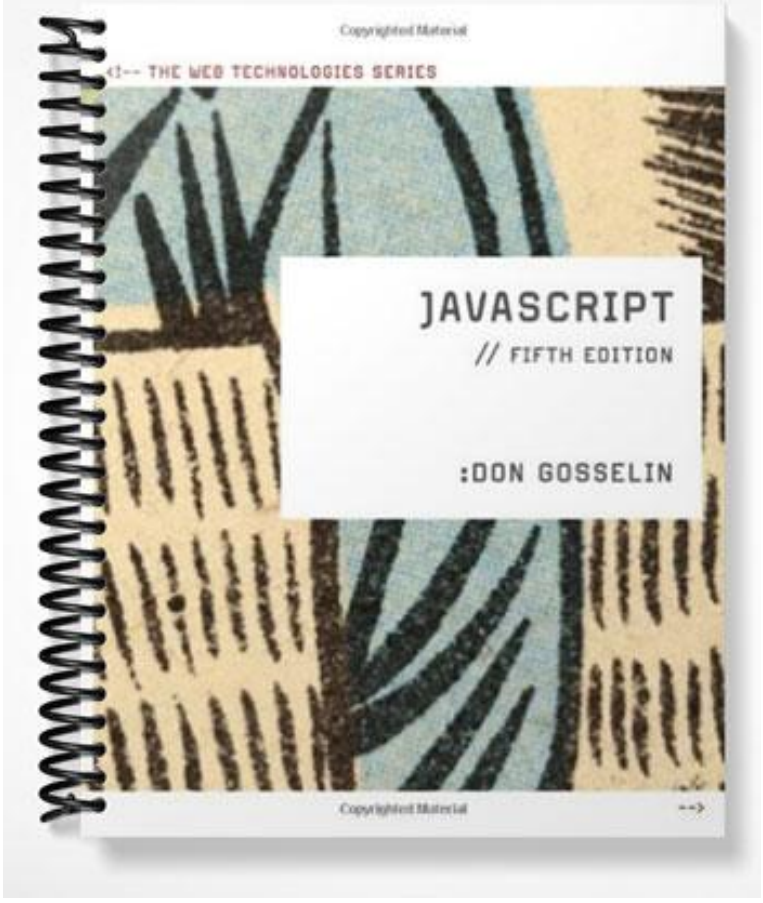
SOLUTIONS MANUAL



Chapter 2

Working with Functions, Data Types, and Operators

At a Glance

Instructor's Manual Table of Contents

Lecture Notes

Overview

So far, the code your students have written has consisted of simple statements placed within script sections. However, almost all programming languages, including JavaScript, allow you to group programming statements in logical units. Students learn how to use functions, groups of statements executed as a single unit, in this chapter. In addition to functions, one of the most important aspects of programming is the ability to store values in computer memory and to manipulate those values. The values, or data, contained in variables are classified into categories known as data types. In this chapter, students will learn about JavaScript data types and the operations that can be performed on them.

Objectives

In this chapter, your students will:

- Learn how to use functions to organize your JavaScript code
- Study data types
- Use expressions and operators
- Study operator precedence

Teaching Tips

Working with Functions

1. Explain that functions are procedures similar to the methods associated with an object. Functions make it possible to treat a related group of JavaScript statements as a single unit.

2. Mention that functions must be contained within a `<script>` element.

Defining Functions

1. Define a function definition as the lines that make up a function.

2. Show the general syntax for defining a function.

3. Define a parameter as a variable that is used within the function and placed within the parentheses that follow a function name.

4. Note that functions can contain multiple parameters separated by commas and provide an example.

5. Describe function statements as the statements that do the actual work of the function (such as calculating the square root of the parameter, or displaying a message on the screen) and note that they must be contained within the function braces. Provide an example.

6. Explain that the student should always put his or her functions within the document head and place calls to a function within the body section. If the program attempts to call a function before it has been created, an error will be generated.

7. Allow time for students to complete the exercise on Page 75 involving the CV Wedding Hall guest book page.

Calling Functions

1. Mention that to execute a function, you must invoke, or call, it.

2. Define a function call as code that calls a function. It consists of the function name, followed by parentheses containing any variables or values to be assigned to the function parameters.

3. Define arguments or actual parameters as variables or values that you place in the parentheses of the function call statement.

4. Define passing arguments as sending arguments to the parameters of a called function.

5. Allow time for students to complete the exercise on Page 76 to add a function call to the `onclick` event handler of the Add Guest button. Use Figure 2-1 to illustrate the guest book page after adding a guest.

6. Mention that sometimes, a student may want a program to receive the results from a called function and then use those results in other code. Provide several examples.

7. Define a return statement as a statement that returns a value to the statement that called the function. Show some examples of a function that returns a value.

8. Allow time for students to complete the exercise on Page 78 to call the `addGuest()` function.

Understanding Variable Scope

1. Mention that a variable's scope defines where a declared variable can be used in a program. Explain that variable scope can be either global or local.

2. Define a global variable as one that is declared outside a function and is available to all parts of your program.

3. Explain that a local variable is declared inside a function and is only available within the function in which it is declared.

4. Mention that you must use the `var` keyword when you declare a local variable, but its use is optional for global variables.

5. Explain why always using the `var` keyword when declaring variables is considered a good programming technique.

6. Explain why declaring a global variable inside of a function by not using the `var` keyword is considered a poor programming technique.

7. Review the script on Page 80 illustrating local and global variable use.

8. Explain that if you declare a variable within a function and do not include the `var` keyword, the variable automatically becomes a global variable.

9. Explain that when a program contains a global variable and a local variable with the same name, the local variable takes precedence when its function is called. Use the script on Pages 80-81 and Figure 2-2 to illustrate this concept.

10. Allow time for students to complete the exercise on Page 81 to add global variables to the index.html page of the CV Wedding Hall site.

Teaching Tip	More information on JavaScript variable scope may be found at: http://javascript.about.com/library/bltut07.htm

Using Built-in JavaScript Functions

1. Use Table 2-1 to describe some of the built-in JavaScript functions as explained in this section.

Teaching Tip	More information on JavaScript functions may be found at: www.w3schools.com/js/js_functions.asp

Quick Quiz 1

1. The lines that make up a function are called the ____.
   Answer: function definition

2. Sending arguments to the parameters of a called function is called ____.
   Answer: passing arguments

3. A(n) ____ statement is a statement that returns a value to the statement that called the function.
   Answer: return

4. A(n) ____ variable is one that is declared outside a function and is available to all parts of your program.
   Answer: global

5. A(n) ____ variable is one that is declared inside a function and is only available within the function in which it is declared.
   Answer: local

Working with Data Types

1. Define a data type as the specific category of information that a variable contains.

2. Define primitive types as data types that can be assigned only a single value. Use Table 2-2 to identify the JavaScript primitive types.

3. Explain the difference between the `null` data type and the `undefined` data type.

4. Refer to the code on Page 84 and Figure 2-3 to illustrate the use of an undefined variable.

5. Explain the difference between a strongly typed programming language and a loosely typed programming language. Mention that JavaScript is a loosely typed programming language.

6. Refer to the code on Page 85 to illustrate how a variable's data type changes automatically each time the variable is assigned a new literal value.

Understanding Numeric Data Types

1. Mention that JavaScript supports two numeric data types: integers and floating-point numbers.

2. Define an integer as a positive or negative number with no decimal places.

3. Define a floating-point number as a number that contains decimal places or is written in exponential notation.

4. Explain that exponential notation, or scientific notation, is a shortened format for writing very large numbers or numbers with many decimal places.

5. Allow time for students to complete the exercise on Page 85 to create a script that prints metric prefixes. Use Figure 2-4 to illustrate how the document looks in a Web browser.

Using Boolean Values

1. Define a Boolean value as a logical value of true or false.

2. Mention that in JavaScript programming, you can only use the words true and false to indicate Boolean values.

3. Refer to the code on Page 89 to illustrate a simple example of two variables that are assigned Boolean values: one true and the other false. Use Figure 2-5 to show the output in a Web browser.

4. Allow time for students to complete the exercise on Page 89 to add five Boolean global variables to the index.html page of the CV Wedding Hall site.

Working with Strings

1. Remind students that a text string is text that is contained within double or single quotation marks.

2. Mention that you can use text strings as literal values or assign them to a variable.

3. Point out that literal strings can be assigned a zero-length string value called an empty string. Explain why a student might want to assign an empty string to a literal string.

4. Mention that to include a quoted string within a literal string surrounded by double quotation marks, the student should surround the quoted string with single quotation marks.

5. Mention that to include a quoted string within a literal string surrounded by single quotation marks, the student should surround the quoted string with double quotation marks.

6. Refer to the code on Page 91 and Figure 2-6 to illustrate as an example of a script that prints strings.

String Operators

1. Point out that when used with strings, the plus sign (+) is known as the concatenation operator, which is used to combine two strings. Show how the operator works with strings.

2. Mention that the compound assignment operator (+=) can also be used to combine two strings. Show how this operator works with strings.

3. Note that the same symbol - a plus sign - serves as the concatenation operator and the addition operator.

Escape Characters and Sequences

1. Explain why a student must use extra care when using single quotation marks with possessives and contractions in strings.

2. Define an escape character as a character that tells the compiler or interpreter that the character that follows it has a special purpose. In JavaScript, the escape character is the backslash (\).

3. Mention that when you combine the escape character with other characters, the combination is called an escape sequence. Most escape sequences carry out special functions.

4. Use Table 2-3 to show the JavaScript escape sequences.

5. Mention that because the escape character itself is a backslash, the student must use the escape sequence \\ to include a backslash as a character in a string.

6. Refer to the code on Page 93 to illustrate an example of a script containing strings with several escape sequences. Use Figure 2-7 to illustrate the output.

7. Allow time for students to complete the exercise on Page 94 to modify the guest book page so that multiple guests may be added.

Teaching Tip	More information on the JavaScript string object may be found at: www.w3schools.com/js/js_obj_string.asp

Quick Quiz 2

1. A(n) ____ is the specific category of information that a variable contains.
   Answer: data type

2. A(n) ____ is a positive or negative number with no decimal places.
   Answer: integer

3. True or False: Empty strings are valid values for literal strings.
   Answer: True

4. True or False: You can use the compound assignment operator (+=) to combine two strings.
   Answer: True

5. In JavaScript, the escape character is the ____.
   Answer: backslash (\)

6. When you combine the escape character with other characters, the combination is called a(n) ____.
   Answer: escape sequence

Using Operators to Build Expressions

1. Use Table 2-4 to discuss the operator types that can be used with JavaScript.

2. Explain the difference between a binary operator and a unary operator.

Arithmetic Operators

1. Mention that arithmetic operators are used in JavaScript to perform mathematical calculations, such as addition, subtraction, multiplication, and division.

Arithmetic Binary Operators

1. Use Table 2-5 to show the arithmetic binary operators available in JavaScript.

2. Refer to the code on Page 97 to illustrate examples of expressions that include arithmetic binary operators. Use Figure 2-8 shows how the expressions appear in a Web browser:

3. Mention that that when JavaScript performs an arithmetic calculation, it performs the operation on the right side of the assignment operator, and then assigns the value to a variable on the left side of the assignment operator.

4. Explain the difference between the division (/) operator and the modulus (%) operator. Refer to the code on Page 99 to illustrate the division (/) operator and the modulus (%) operator. Use Figure 2-9 to illustrate the results.

5. Mention that a student can include a combination of variables and literal values on the right side of an assignment statement and provide examples.

6. Note that a student cannot you cannot include a literal value as the left operand because the JavaScript interpreter must have a variable to which to assign the returned value.

7. Describe how the JavaScript interpreter attempts to convert the string values to numbers when performing arithmetic operations on string values and provide examples.

8. Mention that the JavaScript interpreter does not convert strings to numbers when a programmer uses the addition operator.

Arithmetic Unary Operators

1. Use Table 2-6 to show the arithmetic unary operators available in JavaScript.

2. Explain the difference between a prefix operator and a postfix operator.

3. Explain when to use arithmetic unary operators and provide examples.

4. Use Figure 2-10 and Figure 2-11 to illustrate a simple script that uses the prefix increment operator.

5. Use Figure 2-12 and Figure 2-13 to illustrate a simple script that uses the postfix increment operator.

6. Allow time for students to complete the exercise on Pages 103-105 to modify the index.html file of CV Wedding Hall site so that it calculates the cost of a wedding. Use Figure 2-14 to illustrate the results.

Teaching Tip	More information on JavaScript operators - string and arithmetic operators may be found at: www.webdevelopersnotes.com/tutorials/javascript/javascript_string_arthimetic_operators.php3

Assignment Operators

1. Mention that assignment operators are used for assigning a value to a variable.

2. Describe compound assignment operators.

3. Use Table 2-7 to discuss the assignment operators available in JavaScript.

4. Explain how to use the += compound addition assignment operator to combine two strings and to add numbers. Point out that a value of "NaN" stands for "Not a Number" and is returned when a mathematical operation does not result in a numerical value.

5. Refer to the code on Pages 106-107 to illustrate examples of the different assignment operators.

Comparison and Conditional Operators

1. Define comparison operators as operators used to compare two operands, and to determine if one numeric value is greater than the other. Note that a Boolean value of true or false is returned after two operands are compared.

2. Use Table 2-8 to point out the JavaScript comparison operators.

3. Mention that a programmer can use number or string values as operands with comparison operators. When two numeric values are used as operands, the JavaScript interpreter compares them numerically.

4. Point out that the conditional operator executes one of two expressions, based on the results of a conditional expression.

5. Describe the syntax for the conditional operator.

6. Review the code on Page 108 illustrating an example of the conditional operator.

7. Allow time for students to complete the exercise on Pages 108-111 to add fields and code to the Wedding Planner form that allow users to select live music and flowers. Use Figure 2-15 to illustrate the wedding planner page after adding the live music and flowers fields.

Logical Operators

1. Explain that logical operators are used for comparing two Boolean operands for equality. A Boolean value of true or false is returned after two operands are compared.

2. Use Table 2-9 to describe the JavaScript logical operators.

3. Point out that the Or (`||`) and the And (&&) operators are binary operators (requiring two operands), whereas the Not (`!`) operator is a unary operator (requiring a single operand).

4. Note that logical operators are often used with comparison operators to evaluate expressions, allowing a programmer to combine the results of several expressions into a single statement. Review the example code on Pages 112-113.

Teaching Tip	More information on JavaScript comparison and logical operators may be found at: www.w3schools.com/JS/js_comparisons.asp

Special Operators

1. Use Table 2-10 to discuss the special operators in JavaScript. These operators are used for various purposes and do not fit within any other category.

2. Discuss why the `typeof` operator is useful.

3. Describe the syntax of the `typeof` operator. Use Table 2-11 to list the values that can be returned by the `typeof` operator.

Understanding Operator Precedence

1. Explain what is meant by the term operator precedence.

2. Use Table 2-12 to show the order of precedence for JavaScript operators.

3. Define an operator's associativity as the order in which operators of equal precedence execute.

4. Use Figure 2-16 to explain left to right associativity and Figure 2-17 to explain right to left associativity.

Teaching Tip	More information on JavaScript increment and decrement operators - Operator Precedence may be found at: www.webdevelopersnotes.com/tutorials/javascript/javascript_increment_decrement_operators.php3

Quick Quiz 3

1. ____ are variables and literals contained in an expression.
   Answer: Operands

2. True or False: A unary operator requires an operand before and after the operator.
   Answer: False

3. True or False: A prefix operator is placed before a variable.
   Answer: True

4. ____ operators are used for assigning a value to a variable.
   Answer: Assignment

5. ____ operators are used for comparing two Boolean operands for equality.
   Answer: Logical

6. Operator ____ is the order in which operators of equal precedence execute.
   Answer: associativity

Class Discussion Topics

1. What are the similarities between a function and a procedure?

2. Why does there need to be a set order of precedence for the JavaScript operators?

Additional Projects

1. Have each student write a script in which he or she:
   a. Defines a function called `paySalary` that receives the name of an employee and the number of hours worked.
   b. Ensures `paySalary` calculates the salary of an employee as follows: the first 40 hours are paid at $8 and any extra hour (after the first 40 hours) is paid at $12.
   c. Prints onto the screen the name of the employee and the salary for that employee.

2. Have each student write a script in which he or she:
   a. Declares a variable called `employee1` and assign it the value: "Susan Harper."
   b. Declares a variable called `hoursWorked1` and assigns it the value 45.
   c. Prints the variables onto the screen in such a way that the following is displayed on the screen: Susan Harper has worked 45 hours.

Additional Resources

1. Core JavaScript 1.5 Guide
   https://developer.mozilla.org/en/Core_JavaScript_1.5_Guide

2. Client-Side JavaScript Reference:
   http://docs.sun.com/source/816-6408-10/contents.htm

3. JavaScript Tutorial:
   www.w3schools.com/js/default.asp

4. JavaScript Operations on Variables:
   www.functionx.com/javascript/Lesson04.htm

5. JavaScript Tutorial - Variables:
   www.howtocreate.co.uk/tutorials/javascript/variables

Key Terms

- **Arithmetic operators** are used in JavaScript to perform mathematical calculations, such as addition, subtraction, multiplication, and division.
- The variables or values that you place in the parentheses of the function call statement are called **arguments** or **actual parameters**.
- **Assignment operators** are used for assigning a value to a variable.
- **Associativity** is the order in which operators of equal precedence execute.
- A **binary operator** requires an operand before and after the operator.
- A **Boolean value** is a logical value of true or false.
- To execute a function, you must invoke, or **call**, it from elsewhere in your program.
- **Comparison operators** are used to compare two operands and determine if one numeric value is greater than another.
- **Compound assignment operators** perform mathematical calculations on variables and literal values in an expression, and then assign a new value to the left operand.
- The **concatenation operator (+)** is used to combine two strings.
- The **conditional operator** executes one of two expressions, based on the results of a conditional expression.
- A **data type** is the specific category of information that a variable contains.
- Loose typing is known as **dynamic typing** because data types can change after they have been declared.
- A zero-length string is called an **empty string**.

- An **escape character** tells the compiler or interpreter that the character that follows it has a special purpose.
- When you combine the escape character with other characters, the combination is called an **escape sequence**.
- **Exponential notation**, or **scientific notation**, is a shortened format for writing very large numbers or numbers with many decimal places.
- A **floating-point number** is a number that contains decimal places or that is written in exponential notation.
- In JavaScript programming, you can write your own procedures, called **functions**, which are similar to the methods associated with an object.
- The code that calls a function is referred to as a **function call** and consists of the function name followed by parentheses that contain any variables or values to be assigned to the function parameters.
- The lines that make up a function are called the **function definition**.
- A **global variable** is one that is declared outside a function and is available to all parts of your program.
- An **integer** is a positive or negative number with no decimal places.
- A **local variable** is declared inside a function and is only available within the function in which it is declared.
- **Logical operators** are used for comparing two Boolean operands for equality.
- Programming languages that do not require you to declare the data types of variables are called **loosely typed programming languages**.
- The term **operator precedence** refers to the order in which operations in an expression are evaluated.
- A **parameter** is a variable that is used within a function.
- A **postfix operator** is placed after a variable.
- A **prefix operator** is placed before a variable.
- Sending arguments to the parameters of a called function is called **passing arguments**.
- Data types that can be assigned only a single value are called **primitive types**.
- A **return statement** is a statement that returns a value to the statement that called the function.
- **Exponential notation**, or **scientific notation**, is a shortened format for writing very large numbers or numbers with many decimal places.
- Strong typing is known as **static typing**, because data types do not change after they have been declared.
- Programming languages that require you to declare the data types of variables are called **strongly typed programming languages**.
- A **unary operator** requires a single operand either before or after the operator.
- A **variable's scope** identifies where in a program a declared variable can be used.