

Chapter 2

Working with Functions, Data Types, and Operators

A Guide to this Instructor's Manual:

We have designed this Instructor's Manual to supplement and enhance your teaching experience through classroom activities and a cohesive chapter summary.

This document is organized chronologically, using the same headings that you see in the textbook. Under the headings you will find: lecture notes that summarize the section, Teacher Tips, Class Discussion Topics, and Additional Projects and Resources. Pay special attention to teaching tips and activities geared towards quizzing your students and enhancing their critical thinking skills.

In addition to this Instructor's Manual, our Instructor's Resources also contain PowerPoint Presentations, Test Banks, and other supplements to aid in your teaching experience.

At a Glance

Instructor's Manual Table of Contents

- Overview
- Objectives
- Teaching Tips
- Quick Quizzes
- Class Discussion Topics
- Additional Projects
- Additional Resources
- Key Terms

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. They will also explore the order in which different operations are performed, and how to change that order.

Objectives

After completing this chapter, students will be able to:

- Use functions to organize JavaScript code
- Use expressions and operators
- Identify the order of operator precedence in an expression

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. Discuss the difference between named functions and anonymous functions.
2. Introduce a function definition as the lines of code that create a function.
3. Show the general syntax for defining a function.
4. Define a parameter as a variable that is used within the function and placed within the parentheses that follow a function name.
5. Explain how to define an anonymous function.

6. Note that functions can contain multiple parameters separated by commas, and provide an example.
7. 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.
8. Explain that it is common practice to place functions in an external .js file and include a script section to reference the file at the bottom of the body section of an HTML document.
9. Allow time for students to complete the exercise on Pages 77-79 involving an estimate for photography services for Fan Trick Fine Art Photography.

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. Give an example.

Handling Events with Functions

1. Explain that functions can be called in response to browser events. List the three different methods: HTML attributes, object properties, and event listeners.
2. Describe the simplest way to specify a function as an event handler by specifying the function at the value for the HTML attribute. Provide an example.
3. Explain the alternate method of specifying a function name as the attribute value.
4. Note that most developers prefer not to mix HTML and JavaScript code in the same file.
5. Explain two other ways to specify functions as event handlers: specifying the function as a property value for the object, and using the `addEventListener()` method.

- Describe the major drawback of the first method, that it is limited to only one event handler per event.
6. Give an example of adding an event listener, using the code on the top of Page 82.
 7. Allow time for students to specify the `resetForm()` function as an event handler, following the steps on Pages 82-83.

Locating Errors with the Browser Console

1. Discuss the inevitability of unintentional errors in code. Explain that errors will cause a browser to generate an error message in the browser console.
 - Note that the browser console is hidden from users by default, but can be opened and viewed.
2. Introduce the steps to open the browser console in the major browsers.
3. Allow time for students to practice viewing and correcting an error message in the browser console, following the steps on Pages 83-86.

Using Return Statements

1. Introduce the concept of returning a value from a function to a calling statement. Give an example of a situation in which this would be useful.
 - Note that functions do not necessarily have to return a value.
2. Define a return statement and illustrate its use with the code at the bottom of Page 86.

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 that if you declare a variable within a function and do not include the `var` keyword, the variable automatically becomes a global variable.
7. Explain why declaring a global variable inside of a function by not using the `var` keyword is considered a poor programming technique.
8. Review the scripts on Page 88 illustrating local and global variable use.
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 Page 89 and Figure 2-6 to illustrate this concept.
10. Allow time for students to complete the exercise on Pages 89-90 to add global variables to the `ft.js` file.

**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 2-2 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 of code 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 2-3 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 92 and Figure 2-7 to illustrate the use of an undefined variable.
5. Explain the difference between a strongly typed (statically typed) programming language and a loosely typed (duck typed) programming language. Mention that JavaScript is a loosely typed programming language.
6. Refer to the code on Page 93 to illustrate how a variable's data type changes automatically each time the variable is assigned a new literal value.

Working with Numeric Values

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 (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 Pages 94-96 to create a script that prints metric prefixes. Use Figure 2-8 to illustrate how the document looks in a Web browser.

Working with 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 97 to illustrate a simple example of two variables that are assigned Boolean values: one true and the other false. Use Figure 2-9 to show the output in a Web browser.
4. Allow time for students to complete the exercise on Page 98 to add two Boolean global variables to the ft.js file.

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 99 and Figure 2-10 to illustrate as an example of string examples in a browser.

String Operators

1. Point out that when used with strings, the plus sign (+) is called the concatenation operator, and 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 2-4 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 103 to illustrate an example of a script containing strings with several escape sequences. Use Figure 2-11 to illustrate the output.

**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 2-5 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, division and returning the modulus (remainder).

Arithmetic Binary Operators

1. Use Table 2-6 to show the arithmetic binary operators available in JavaScript.
2. Refer to the code on Pages 106-107 to illustrate examples of expressions that include arithmetic binary operators. Use Figure 2-12 to show 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 108 to illustrate the division (/) operator and the modulus (%) operator. Use Figure 2-13 to illustrate the output.
5. Mention that a student can include a combination of variables and literal values on the right side of an assignment statement. Provide examples.
6. Note that a student 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 string values to numbers when performing arithmetic operations on string values. 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 2-7 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. Provide examples.

4. Use the code at the top of Page 111 and Figure 2-14 to illustrate a simple script that uses the prefix increment operator.
5. Use the code starting at the bottom of Page 111 and Figure 2-15 to illustrate a simple script that uses the postfix increment operator.

Teaching Tip

More information on JavaScript operators - string and arithmetic operators may be found at:
www.webdevelopersnotes.com/tutorials/javascript/javascript_string_arithmetic_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 2-8 to review 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 Page 114 to illustrate examples of the different assignment operators.
6. All time for the student to modify the ft.js file to calculate the cost of hiring photography staff, following the steps on Pages 115-118.

Comparison and Conditional Operators

1. Define comparison operators (relational operators) as operators used to compare two operands, and 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 2-9 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 120, illustrating an example of the conditional operator.
7. Allow time for students to complete the exercise on Pages 120-122 to create functions for the estimate form that add the cost of a memory book and reproduction rights. Use

Understanding Falsy and Truthy Values

1. Introduce the six values treated in comparison operations as the Boolean value `false`.
2. Emphasize that all values other than these six falsy values are the equivalent of the Boolean `true`, and are known as truthy values.
3. Illustrate the use of falsy and truthy values, using the code at the top of Page 123.

Logical Operators

1. Explain that logical operators are used to modify Boolean values or specify the relationship between operands in an expression that results in a Boolean value.
2. Use Table 2-10 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 2-11 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 2-12 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 2-13 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.
5. Allow time for the students to modify the `calcStaff()` function to include mileage charges, following the steps on Pages 130-131.

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.oracle.com/cd/E19957-01/816-6408-10/>
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.howtcreate.co.uk/tutorials/javascript/variables

Key Terms

- **actual parameters**—*See* arguments
- **adding an event listener**—Specifying an event handler with the `addEventListener()` method
- **anonymous function**—A set of related statements with no name assigned to it
- **arguments**—The variables or values that you place in the parentheses of a function call statement

- **arithmetic operators**—Operators used to perform mathematical calculations, such as addition, subtraction, multiplication, and division
- **assignment operator**—An operator used for assigning a value to a variable
- **associativity**—The order in which operators of equal precedence execute
- **binary operator**—An operator that requires an operand before and after it
- **Boolean value**—A logical value of true or false
- **browser console**—A browser pane that displays error messages
- **call**—To invoke a function from elsewhere in your code
- **comparison operator**—An operator used to compare two operands and determine if one value is greater than another
- **compound assignment operators**—Assignment operators other than the equal sign, which perform mathematical calculations on variables and literal values in an expression, and then assign a new value to the left operand
- **concatenation operator**—The plus sign (+) when used with strings; this operator combines, or concatenates, string operands
- **conditional operator**—The `?:` operator, which executes one of two expressions based on the results of a conditional expression
- **console**—*See* browser console
- **data type**—The specific category of information that a variable contains, such as numeric, Boolean, or string
- **duck typed**—*See* loosely typed
- **dynamically typed**—*See* loosely typed
- **empty string**—A zero-length string value
- **escape character**—The backslash character (`\`), which tells JavaScript compilers and interpreters that the character that follows it has a special purpose
- **escape sequence**—The combination of the escape character (`\`) with one of several other characters, which inserts a special character into a string; for example, the `\b` escape sequence inserts a backspace character
- **exponential notation**—A shortened format for writing very large numbers or numbers with many decimal places, in which numbers are represented by a value between 1 and 10 multiplied by 10 raised to some power
- **falsy values**—Six values that are treated in comparison operations as the Boolean value `false`
- **floating-point number**—A number that contains decimal places or that is written in exponential notation
- **function**—A related group of JavaScript statements that are executed as a single unit
- **function braces**—The set of curly braces that contain the function statements in a function definition
- **function call**—The code that calls a function, which consists of the function name followed by parentheses, which contain any arguments to be passed to the function
- **function definition**—The lines that make up a function
- **function statements**—The statements that do the actual work of a function, such as calculating the square root of the parameter, or displaying a message on the screen, and which must be contained within the function braces
- **global variable**—A variable that is declared outside a function and is available to all parts of a program, because it has global scope

- **innerHTML property**—The property of a web page object whose value is the content between the element's opening and closing tags
- **integer**—A positive or negative number with no decimal places
- **local variable**—A variable that is declared inside a function and is available only within the function in which it is declared, because it has local scope
- **logical operators**—The Or (||), And (&&), and Not (!) operators, which are used to modify Boolean values or specify the relationship between operands in an expression that results in a Boolean value
- **loosely typed**—Description of a programming language that does not require you to declare the data types of variables
- **named function**—A set of related statements that is assigned a name
- **operator precedence**—The system that determines the order in which operations in an expression are evaluated
- **parameter**—A variable that is used within a function
- **passing arguments**—Sending arguments to the parameters of a called function
- **postfix operator**—An operator that is placed after a variable name
- **prefix operator**—An operator that is placed before a variable name
- **primitive types**—Data types that can be assigned only a single value
- **relational operator**—*See* comparison operator
- **return statement**—A statement in a function that returns a value to the statement that called the function.
- **scientific notation**—*See* exponential notation
- **statically typed**—*See* strongly typed
- **strongly typed**—Description of a programming language that requires you to declare the data types of variables
- **textContent property**—A property similar to the innerHTML property, except that its value excludes any HTML markup
- **truthy values**—All values other than the six falsy values; truthy values are treated in comparison operations as the Boolean value true
- **unary operator**—An operator that requires just a single operand either before or after it.
- **variable scope**—The aspect of a declared variable that determines where in code it can be used, either globally (throughout the code) or locally (only within the function in which it is declared)