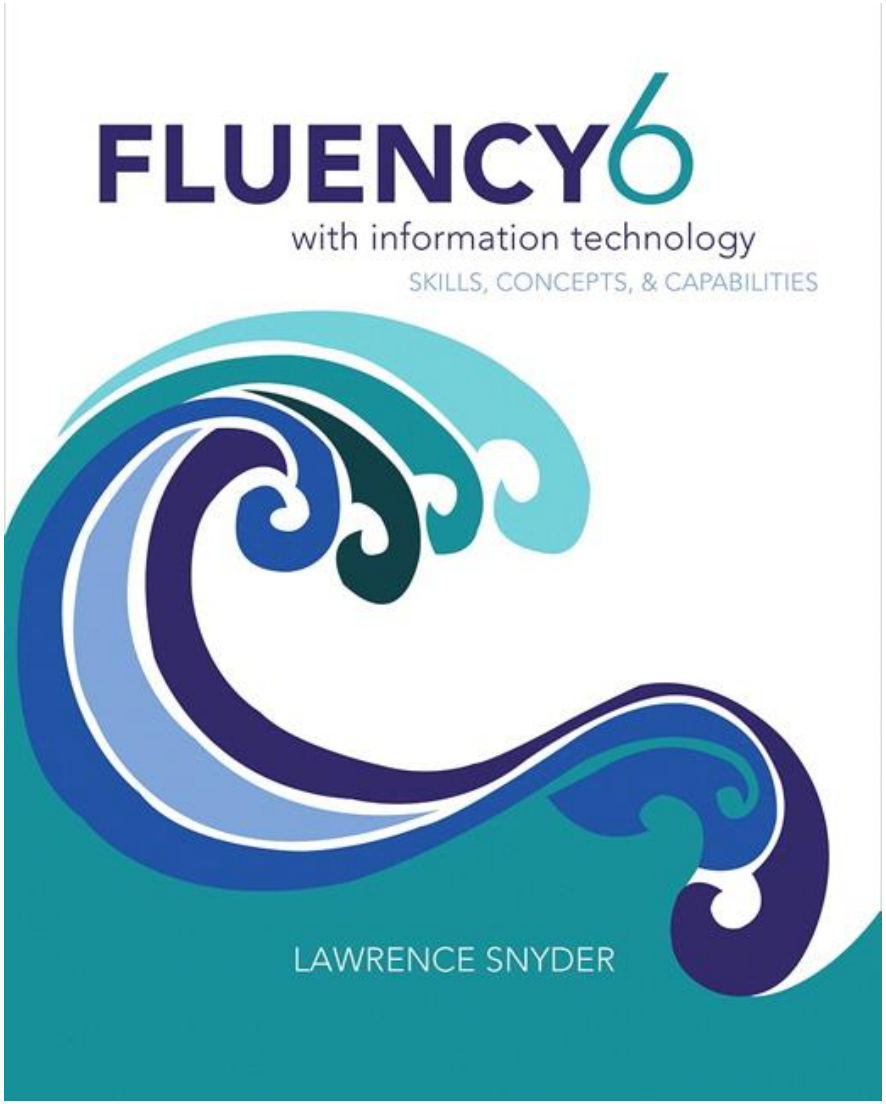


Chapter 17

Fundamental Concepts Expressed in JavaScript

Learning Objectives

- Tell the difference between name, value, and variable
- List three basic data types and the rules for specifying them in a program
- Explain the way in which the assignment statement changes a variable's value
- Write expressions using arithmetic, relational, and logical operators
- Write conditional and compound statements

Programming Concepts

- Programming is the act of formulating an algorithm or program
- A systematic means of solving a problem
  - Someone (and a computer) can follow the instructions and produce the intended result for *every* input, *every* time

Programming Concepts

- The program must be able to perform or be executed without the programmer
- All steps ***must*** be spelled out precisely and effectively
- All contingencies must be planned for

Programming Concepts

- Programming requires thinking
- Basic programming concepts provide tools needed to formulate *any* computation
- Trying to program an algorithm precisely using English is hopeless
- Natural languages are too ambiguous for directing anything as clueless as a computer

Programming Concepts

- This chapter introduces the following programming concepts:
  - Names, values, and variables
  - Declarations
  - Data types, numbers, string literals, and Booleans
  - Assignment
  - Expressions
  - Conditionals

Names, Values, and Variables

- Names Have Changing Values
  - Names and values are separable in programming
  - Names have changing values

Name	Current Value (1/20/14)	Previous Values
U.S. President	Barack Obama	Bill Clinton, George H. W. Bush
Chief Justice U.S. Supreme Court	John Roberts	Warren Burger, Earl Warren
James Bond	Daniel Craig	Sean Connery, Roger Moore
Queen of England	Elizabeth II	Victoria I, Elizabeth I
U.N. Secretary General	Ban Ki-moon	Boutros Boutros-Ghali, Kofi Annan

Names, Values, and Variables

- Names change values
- A program is a fixed specification of a process
- As the process evolves, the names must refer to these new values
- In computing, the name is always separable from the value, and the value can be changed

Names in a Program are Called Variables

- In programming terminology, the names are called ***variables***
- Variables mean that values *vary*
- The most commonly used programming language operation is the command to change the value of a variable:
  - Called ***assignment***

Identifiers and Their Rules

- The letter sequence that makes up a variable's name is called the ***identifier***
- Identifiers have a particular form
- Identifiers must begin with a letter, followed by any sequence of letters, numerals, or the underscore_symbol
- Identifiers are not allowed to contain spaces

Identifiers and Their Rules

- Note two features of identifiers:
  - The underscore symbol can be used as a word separator
    - It makes identifiers more readable
    - “No spaces” rule
  - Identifiers are case sensitive
    - UPPERCASE and lowercase letters are different

Variable Declaration Statement

- Programs are usually written “starting from scratch”
- The first thing to do when writing any program is to state or ***declare*** the variables that will be used
- ***Declaring variables*** is done using a command called a ***declaration***
- In JavaScript, the declaration command is the word ***var***, followed by a list of the identifiers for the variables to be declared, separated by commas

Variable Declaration Statement

var area, radius;

- This *command* declares that two identifiers (area, radius) will be used as variables

The Statement Terminator

- A program is simply a list of statements;
- Each statement must be terminated by some punctuation symbol;
- The statement terminator in JavaScript is the semicolon;
- The computer needs the semicolon to know when a statement is complete;
- Terminate every statement with a semicolon;

Rules for Declaring Variables

- Every variable used *must* be declared
- JavaScript allows declaration statements anywhere in the list of statements
- Variable declarations announce what variables will be used in the program
- Declare variables first

Undefined Values

- The declaration states that the identifier is the *name* of a variable
- The name has no value at first, it is not defined
- It is a name that doesn't name anything
- The name is declared but there is no value assigned yet
  - The value is ***undefined***

Initializing a Declaration

- Sometimes there is an initial value for identifiers
  - JavaScript allows setting the initial value as part of the declaration
  - This is called ***initializing*** the variable
- Declaring variables with initial values is written as:
  - *var taxRate = .088;*
  - *var balanceDue = 0;*

Initializing a Declaration

- Variables can be declared and initialized by separating them with commas:
  - *var taxRate = .088, balanceDue = 0;*
- Usually several variables are declared in a single declaration statement when the variables are logically related
- If the variables are not related, they are usually specified in separate statements

Three Basic *Data Types* of JavaScript

- There are three types of data in the JavaScript programs used in this book:
  1. numbers,
  2. strings, and
  3. Booleans

Rules for Writing Numbers

- There are rules for writing numbers
- One “unusual” aspect of numbers in programming is that there are no “units”
  - Numbers must be written in decimal form (0.33, not 33%; 10.89, not $10.89)
- Standard computer numbers
  - Have about 10 significant digits
  - Range from as small as 10^{-324} to as large as 10^{308}

Rules for Writing Numbers

- Numbers and computer arithmetic are unexpectedly subtle
- As a general rule, the “safe zone” for numbers is the range from 2 billionths to 2 billion plus or minus

Strings

- Strings are a common kind of data
- ***Strings*** are “sequences of keyboard characters”
- Notice that a string is always surrounded by single (') or double (") quotes
- Strings can initialize a declaration
- Strings are needed when manipulating text

Rules for Writing Strings in JavaScript

- There are rules for writing strings in JavaScript:
  - Strings must be surrounded by quotes, either single (') or double ("), which are *not* curly
  - Most characters are allowed within quotes except:
    - the return character (

), backspace character, tab character, \, and two little used others
  - Double quoted strings can contain single quotes, and vice versa

Rules for Writing Strings in JavaScript

- There are rules for writing strings in JavaScript:
  - The apostrophe (') is the same as the single quote
  - Any number of characters is allowed in a string.
  - The minimum number of characters in a string is zero (""), which is called the ***empty string***

Strings

- To use double quotes in a string, enclose the string in single quotes:
  var answer = 'He said, "No!“ ‘
- If our string contains single quotes, enclose it in double quotes:
  var book = "Guide to B&B's"
- Since the apostrophe is commonly used in possessives and contractions, use double quotes as the default

String constants or string literals

- The term ***literal*** means that the characters are typed *literally* in the program
- There are rules about how to write literals:
  - The surrounding quotes are removed when the literal is stored in the computer
  - Any character can be stored in the computer's memory
  - Prohibited characters can be the value of a string in the computer by using the "escape" mechanism

Escape Mechanisms

- For JavaScript, the escape symbol is the backslash (\)
- The escape sequences are converted to the single characters they represent when stored in the computer's memory

Table 17.1 Escape sequences for characters prohibited from string literals.

Sequence	Character	Sequence	Character
\b	Backspace	\f	Form feed
\n	New line	\r	Carriage return
\t	Tab	\'	Apostrophe or single quote
\"	Double quote	\\	Backslash

Boolean Values

- Another kind of value is the Boolean value (Booleans)
- There are only two Boolean values: *true* and *false*
- Boolean values are written as letter sequences, they are values, not identifiers or strings
- Booleans are used implicitly throughout the programming process

Values in Programming

- The different kinds of values of a programming language are called its ***data types***
- There are three types used here for JavaScript: numbers, strings, and Booleans
- There are several other types

The Assignment Statement

- If variables are to change values in an algorithm or program, there should be a command to do so
- The ***assignment statement*** changes a variable's value
- An assignment statement has three parts that always occur in this order:

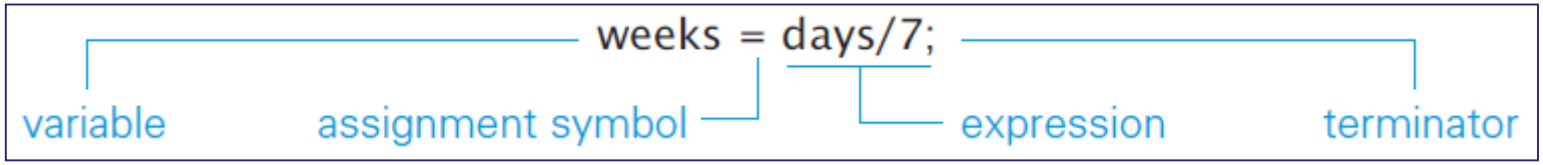
  <variable> <assignment symbol> <expression>;

The Assignment Statement

- The assignment statement is terminated by a semicolon
- JavaScript's <assignment symbol> is the equal sign (=)

Assignment Symbol

- Different programming languages use different symbols for indicating assignment
- The three most widely used symbols are:
  - The equal sign (=)
  - The colon/equal sign pair (:=)
  - The left pointing arrow (←)
  - An example of an assignment is:



Interpreting an Assignment Statement

- To understand how assignment works, you must think of a value flowing from the right side (expression side) to the left side (variable side)
- The assignment symbol should be read as "*is assigned* " or "*becomes*" or "*gets*"
- These terms emphasize the role that the assignment symbol plays in changing the value of the variable named

Interpreting an Assignment Statement

- In an assignment statement everything to the right of the assignment symbol is computed/evaluated first
- If there are any variables used, their current values are used
- The evaluation produces a value that then becomes the new value of the variable named on the left side

Interpreting an Assignment Statement

- Executing the sample assignment statement: weeks = days/7;
- The current value of days is retrieved from memory
- That value is divided by 7
- The answer becomes the new value of the variable weeks

Three Key Points About Assignment

- There are three key points to remember about assignment statements:
  1. All three of the components must be given
  2. The flow of the value to the name is always right to left.
  3. The values of any variables used in the expression are their values *before* the start of execution of the assignment
- This last point is the most important

Right Side First, Then Assign

- It is important that the right side is evaluated before the value is assigned
- This allows a variable to be updated:

  totalScore = totalScore + 3
- An assignment is *not* a comparison

Lab Practice

- From now on in our JavaScript study, everything we learn, we can try in the Firefox browsers
  - learning programming is active!
- We will use Firefox Scratchpad
- Then transfer the JavaScript to other Web pages

Lab Practice

- Tools > Web Developer > Scratchpad
- Comments in JavaScript
  - multiline begin with /* and end with */
  - end of line use //
- Type in the code below

```
var days = 77;
var weeks;
weeks = days / 7;
```

Lab Practice

- To see the result, use the Display command (^L)
- Change days to be initialized to some other number, instead of 77 and display the new result

Lab Practice

- Initialize two more variables (totalScore, and shotClock) to 10
- Then write two assignment statements, one that adds 3 to totalScore and the other that subtracts 1 from shotClock
- To Save your work type ^S, then save the file with a .js extension

An Expression and Its Syntax

- Programming is not mathematics but it has its roots there
- One programming concept is an algebra-like formula called an ***expression***
- Expressions describe the means of performing an actual computation
- Expressions are built of variables and operators:
  - addition (+) and subtraction(–)
  - multiplication (*) and division (/)
- These are called the ***arithmetic operators***

Arithmetic Operators

- Expressions *usually* follow rules similar to algebraic formulas
- Multiplication must be given explicitly with the asterisk (*) multiply operator: *a * b*
- Multiplication and division are performed before addition and subtraction
  - They have a higher precedence than addition and subtraction
  - Parentheses can bypass that order

Arithmetic Operators

- Superscripts (x^2) are prohibited
- Some languages have an operator for exponents or powers, but not JavaScript
- If we want to square the value of *x*, then you must multiply it times itself:
  x * *x*

Arithmetic Operators

- Operators like + and * are called ***binary operators***
  - They operate on two values
  - The values are called ***operands***
- There are also unary operators
  - Negate (−) has only one operand
  - This is NOT subtract

Arithmetic Operators

- Another useful operator is ***mod***
- The modulus (mod) operation (*%*) divides two integers and returns the *remainder*
  - The result of a % b for integers a and b is the remainder of the division a/b
  - Examples:
    4%2 is 0 because 2 evenly divides 4
  - 5%2 is 1 because 2 into 5 leaves a remainder of 1

Relational Operators

- ***Relational operators*** make *comparisons* between numerical values
- The relationship between two numbers is tested
- The outcome of the comparison is a Boolean value of *true* or *false*

a < b	Is a less than b?
a <= b	Is a less than or equal to b?
a == b	Is a equal to b?
a != b	Is a not equal to b?
a >= b	Is a greater than or equal to b?
a > b	Is a greater than b?

- The “equal to” relational operator (==) is a double equal sign
- The “not equal to” operator uses the !

Logical Operators

- The relational test results in a true or false outcome
- Either the two operands are related the indicated way, or they are not
- It is common to test two or more relationships together
  - This requires that relational expression results be combined

Logical Operators

- ***Logical and***
  - The && is the *logical and* operator
  - It plays the same role AND plays in query expressions
  - The outcome of ***a && b*** is true if both a **and** b are true; otherwise, it is false
  - The operands a and b can be variables, or expressions, or a mix

Logical *and*

Value of age	age > 12	age < 20	age > 12 && age < 20
4	false	true	false
16	true	true	true
50	true	false	false

Logical Operators

- ***Logical or***
  - The outcome of **a || b** is true if:
    - either a is true *or* b is true
    - if they are both true
  - It is false only if *both* are false
- && and || have lower precedence than the relational operators
  - Relationals are always tested first
  - It doesn't matter how the operands of || are produced; it only matters that they are true or false values.

Logical ***not***

- ***logical not (!)***
  - It is a unary operator, taking only a single operand
  - Its outcome is the *opposite* of the value of its operand
  - By placing the logical not operator in front of the parenthesized expression, we have a new expression with the opposite outcome

Operator Overload

- ***Operator overload*** is a technical term meaning the "use of an operator with different data types"
  - Strings to numbers or to Booleans
- Operators usually apply to a single data type
  - 4 **+** 5 produces the numerical result of 9
- If operands are strings, what does the **+** mean?

Concatenation

- When we use + with strings, it joins the strings by the operation of ***concatenation***
- It just means that the two strings are placed together if we want them joined
- The meaning of + is overloaded:
  - + to mean addition when operands are numeric
  - + means concatenation when the operands are strings

Conditional Statements

- A conditional statement or a ***conditional*** makes testing numbers and strings simple
- The conditional has the form:
  if (***<Boolean expression>***)
  <then-statement>;
- The ***<Boolean expression>*** is an expression evaluating to true or false
- The ***<then-statement>*** is any JavaScript statement

if Statements and Their Flow of Control

```
if (waterTemp < 32)
   waterState = "Frozen";
```

- The <Boolean expression> is called a predicate
- It is evaluated, resulting in a true or false outcome
- If the outcome is *true*, the <then-statement> is performed
- If the outcome is *false*, the <then-statement> is skipped

if Statements and Their Flow of Control

```
if (waterTemp < 32)
   waterState = "Frozen";
```

- Writing the <then-statement> indented on the following line is common practice
- Programmers write the <then-statement> indented on the following line to set it off
- The indent emphasizes its conditional nature

Compound Statements

- Programming languages allow for a sequence of statements in the <then-statement>
- Group multiple statements by surrounding them with “curly braces” { }
- { } collects single statements together to become a compound statement

Compound Statements

- The opening **{** is placed immediately *after* the predicate to signal that a compound statement is next
- The closing **}** is placed conspicuously on its own line after the last statement within the compound
- The closing **}** should not be followed by a semicolon

if/else Statements

- The **if/else statement** contain statements that will be executed when the condition's outcome is false

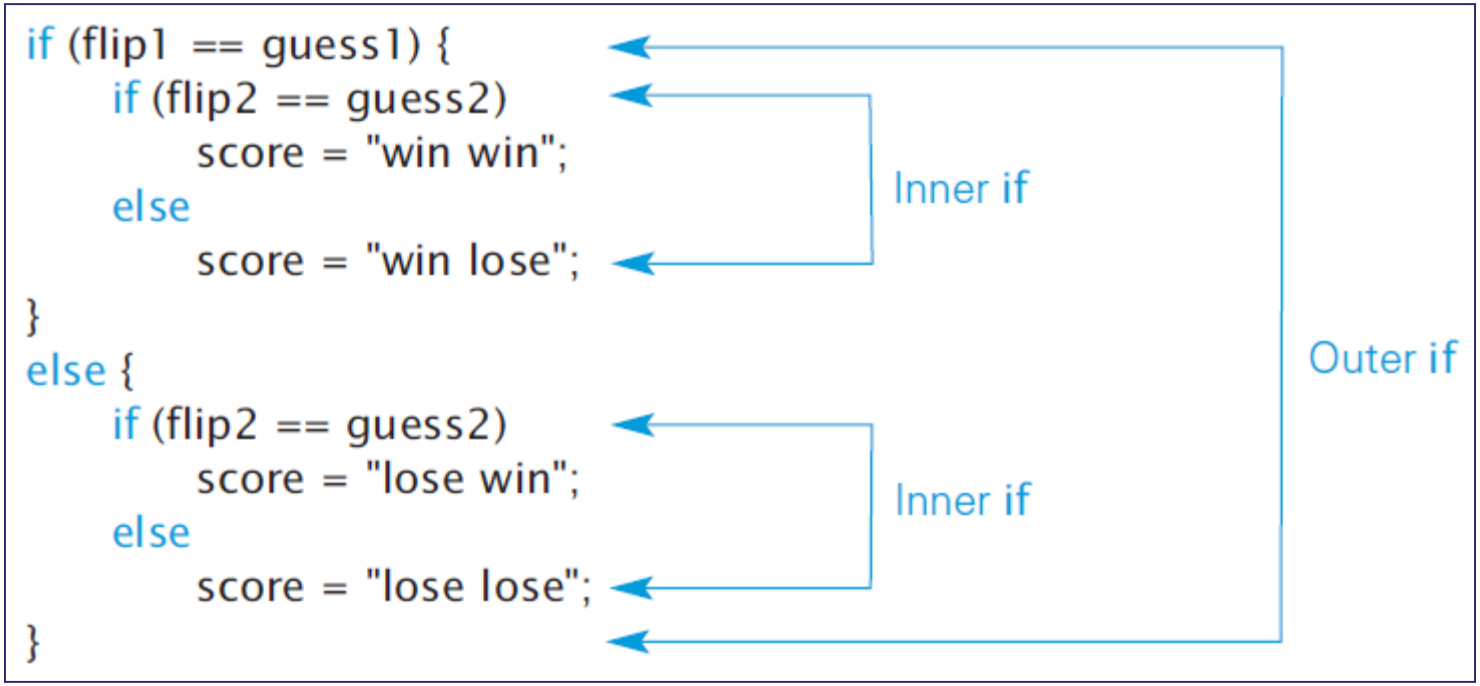
```
if (<Boolean expression>)
   <then-statement>;
else
   <else-statement>;
```

if/else Statements

- The <Boolean expression> is evaluated first
  - If the <Boolean expression>'s outcome is *true*:
    - The <then-statement> is executed
    - The <else-statement> is skipped
  - If the <Boolean expression>'s outcome is *false*:
    - The <then-statement> is skipped
    - The <else-statement> is executed

Nested **if/else Statements**

- Both the ***<then-statement>*** and the ***<else-statement>*** can contain an i***f/els***e
- The rule in JavaScript and most other programming languages is that the ***else*** associates with the (immediately) preceding ***if***
- This can be confusing to read
- The best policy is to enclose the ***<then-statement>*** or ***<else-statement>*** in compound curly braces whenever they contain an if/else


if (flip1 == guess1) {
if (flip2 == guess2)
score = "win win";
else
score = "win lose";
}
else {
if (flip2 == guess2)
score = "lose win";
else
score = "lose lose";
}
Inner if
Inner if
Outer if

The Espresso Program

- The program computes the price of four kinds of espresso drinks based on:
  - The type of drink
  - The size of drink
  - The number of additional shots
  - Plus tax

```
1  /* Espresso Pricing Program
2  Inputs:
3     drink can be "espresso", "latte",
4                  "cappuccino" or "Americano"
5     ounce can be 8, 12, 16
6     shots can be 1, 2, 3, 4, ...
7  Output:
8     price in dollars                          */
9
10 var drink = "latte";
11 var ounce = 12;
12 var shots = 2;
13 var price;
14 var taxRate = 0.088;
15
16 if (drink == "espresso")
17   price = 1.40;
18 if (drink == "latte" || drink == "cappuccino") {
19   if (ounce == 8)
20      price = 1.95;
21   if (ounce == 12)
22      price = 2.35;
23   if (ounce == 16)
24      price = 2.75;
25 }
26 if (drink == "Americano")
27   price = 1.20 + 0.30*(ounce/8);
28
29 price = price + (shots - 1)*.50;
30 price = price + price*taxRate;
31
32 /*
33 3.1008
34 */
```

Figure 17.3 The Espresso Pricing Program run using Scratchpad.

The Espresso Program

- The input and output variables are listed in the initial comment, lines 2-8.
- The input variables are assigned on lines 10-12.
- The program will assign the output value

```
1  /* Espresso Pricing Program
2  Inputs:
3     drink can be "espresso", "latte",
4                  "cappuccino" or "Americano"
5     ounce can be 8, 12, 16
6     shots can be 1, 2, 3, 4, ...
7  Output:
8     price in dollars                          */
9
10 var drink = "latte";
11 var ounce = 12;
12 var shots = 2;
13 var price;
14 var taxRate = 0.088;
15
16 if (drink == "espresso")
17   price = 1.40;
18 if (drink == "latte" || drink == "cappuccino") {
19   if (ounce == 8)
20      price = 1.95;
21   if (ounce == 12)
22      price = 2.35;
23   if (ounce == 16)
24      price = 2.75;
25 }
26 if (drink == "Americano")
27   price = 1.20 + 0.30*(ounce/8);
28
29 price = price + (shots - 1)*.50;
30 price = price + price*taxRate;
31
32 /*
33 3.1008
34 */
```

Figure 17.3 The Espresso Pricing Program run using Scratchpad.

The Espresso Program

- The output is declared to be a variable at line 13.
- Lines 16 through 27 determine the kind of drink and establish the base price

```
1  /* Espresso Pricing Program
2  Inputs:
3     drink can be "espresso", "latte",
4                  "cappuccino" or "Americano"
5     ounce can be 8, 12, 16
6     shots can be 1, 2, 3, 4, ...
7  Output:
8     price in dollars                          */
9
10 var drink = "latte";
11 var ounce = 12;
12 var shots = 2;
13 var price;
14 var taxRate = 0.088;
15
16 if (drink == "espresso")
17   price = 1.40;
18 if (drink == "latte" || drink == "cappuccino") {
19   if (ounce == 8)
20      price = 1.95;
21   if (ounce == 12)
22      price = 2.35;
23   if (ounce == 16)
24      price = 2.75;
25 }
26 if (drink == "Americano")
27   price = 1.20 + 0.30*(ounce/8);
28
29 price = price + (shots - 1)*.50;
30 price = price + price*taxRate;
31
32 /*
33 3.1008
34 */
```

Figure 17.3 The Espresso Pricing Program run using Scratchpad.

The Espresso Program

- The following lines are executed with the purchase of a double tall latte:
  1. Line 16 is executed: The test drink == "espresso" fails, because the variable drink has the value "latte“, then line 17 is skipped

```
1  /* Espresso Pricing Program
2  Inputs:
3     drink can be "espresso", "latte",
4                  "cappuccino" or "Americano"
5     ounce can be 8, 12, 16
6     shots can be 1, 2, 3, 4, ...
7  Output:
8     price in dollars                          */
9
10 var drink = "latte";
11 var ounce = 12;
12 var shots = 2;
13 var price;
14 var taxRate = 0.088;
15
16 if (drink == "espresso")
17   price = 1.40;
18 if (drink == "latte" || drink == "cappuccino") {
19   if (ounce == 8)
20      price = 1.95;
21   if (ounce == 12)
22      price = 2.35;
23   if (ounce == 16)
24      price = 2.75;
25 }
26 if (drink == "Americano")
27   price = 1.20 + 0.30*(ounce/8);
28
29 price = price + (shots - 1)*.50;
30 price = price + price*taxRate;
31
32 /*
33 3.1008
34 */
```

Figure 17.3 The Espresso Pricing Program run using Scratchpad.

The Espresso Program

- The purchase of a double tall latte:
  2. Line 18 is executed: The test drink == "latte" || drink == "cappuccino" has a true outcome

```
1  /* Espresso Pricing Program
2  Inputs:
3     drink can be "espresso", "latte",
4                  "cappuccino" or "Americano"
5     ounce can be 8, 12, 16
6     shots can be 1, 2, 3, 4, ...
7  Output:
8     price in dollars                          */
9
10 var drink = "latte";
11 var ounce = 12;
12 var shots = 2;
13 var price;
14 var taxRate = 0.088;
15
16 if (drink == "espresso")
17   price = 1.40;
18 if (drink == "latte" || drink == "cappuccino") {
19   if (ounce == 8)
20      price = 1.95;
21   if (ounce == 12)
22      price = 2.35;
23   if (ounce == 16)
24      price = 2.75;
25 }
26 if (drink == "Americano")
27   price = 1.20 + 0.30*(ounce/8);
28
29 price = price + (shots - 1)*.50;
30 price = price + price*taxRate;
31
32 /*
33 3.1008
34 */
```

Figure 17.3 The Espresso Pricing Program run using Scratchpad.

The Espresso Program

- The purchase of a double tall latte:
  3. Line 19 is executed: The test ounce == 8 has a false outcome, so its then line 20 is skipped
  4. Line 21 is executed: The ounce == 12 test is true, so the then line 22 is executed

```
1  /* Espresso Pricing Program
2  Inputs:
3     drink can be "espresso", "latte",
4                  "cappuccino" or "Americano"
5     ounce can be 8, 12, 16
6     shots can be 1, 2, 3, 4, ...
7  Output:
8     price in dollars                          */
9
10 var drink = "latte";
11 var ounce = 12;
12 var shots = 2;
13 var price;
14 var taxRate = 0.088;
15
16 if (drink == "espresso")
17   price = 1.40;
18 if (drink == "latte" || drink == "cappuccino") {
19   if (ounce == 8)
20      price = 1.95;
21   if (ounce == 12)
22      price = 2.35;
23   if (ounce == 16)
24      price = 2.75;
25 }
26 if (drink == "Americano")
27   price = 1.20 + 0.30*(ounce/8);
28
29 price = price + (shots - 1)*.50;
30 price = price + price*taxRate;
31
32 /*
33 3.1008
34 */
```

Figure 17.3 The Espresso Pricing Program run using Scratchpad.

The Espresso Program

- The purchase of a double tall latte:
  5. Line 23 is executed: The ounce == 16 test fails, so its then statement is skipped
  6. Line 26 is executed: The drink == "Americano" test fails, so its then statement is skipped

```
1  /* Espresso Pricing Program
2  Inputs:
3     drink can be "espresso", "latte",
4                  "cappuccino" or "Americano"
5     ounce can be 8, 12, 16
6     shots can be 1, 2, 3, 4, ...
7  Output:
8     price in dollars                          */
9
10 var drink = "latte";
11 var ounce = 12;
12 var shots = 2;
13 var price;
14 var taxRate = 0.088;
15
16 if (drink == "espresso")
17   price = 1.40;
18 if (drink == "latte" || drink == "cappuccino") {
19   if (ounce == 8)
20      price = 1.95;
21   if (ounce == 12)
22      price = 2.35;
23   if (ounce == 16)
24      price = 2.75;
25 }
26 if (drink == "Americano")
27   price = 1.20 + 0.30*(ounce/8);
28
29 price = price + (shots - 1)*.50;
30 price = price + price*taxRate;
31
32 /*
33 3.1008
34 */
```

Figure 17.3 The Espresso Pricing Program run using Scratchpad.

The Espresso Program

- The purchase of a double tall latte:
  7. Line 29 is executed: This causes the value of shots minus 1 to be multiplied by .50, resulting in the value .50, which is added to price, yielding price of 2.85

```
1  /* Espresso Pricing Program
2  Inputs:
3     drink can be "espresso", "latte",
4                  "cappuccino" or "Americano"
5     ounce can be 8, 12, 16
6     shots can be 1, 2, 3, 4, ...
7  Output:
8     price in dollars                          */
9
10 var drink = "latte";
11 var ounce = 12;
12 var shots = 2;
13 var price;
14 var taxRate = 0.088;
15
16 if (drink == "espresso")
17   price = 1.40;
18 if (drink == "latte" || drink == "cappuccino") {
19   if (ounce == 8)
20      price = 1.95;
21   if (ounce == 12)
22      price = 2.35;
23   if (ounce == 16)
24      price = 2.75;
25 }
26 if (drink == "Americano")
27   price = 1.20 + 0.30*(ounce/8);
28
29 price = price + (shots - 1)*.50;
30 price = price + price*taxRate;
31
32 /*
33 3.1008
34 */
```

Figure 17.3 The Espresso Pricing Program run using Scratchpad.

The Espresso Program

- The purchase of a double tall latte:
  8. Line 30 is executed: Current value of price is multiplied by taxRate, resulting in 0.25, which is added to price to compute the final value of 3.10, which is assigned to price

```
1  /* Espresso Pricing Program
2  Inputs:
3     drink can be "espresso", "latte",
4                  "cappuccino" or "Americano"
5     ounce can be 8, 12, 16
6     shots can be 1, 2, 3, 4, ...
7  Output:
8     price in dollars                          */
9
10 var drink = "latte";
11 var ounce = 12;
12 var shots = 2;
13 var price;
14 var taxRate = 0.088;
15
16 if (drink == "espresso")
17   price = 1.40;
18 if (drink == "latte" || drink == "cappuccino") {
19   if (ounce == 8)
20      price = 1.95;
21   if (ounce == 12)
22      price = 2.35;
23   if (ounce == 16)
24      price = 2.75;
25 }
26 if (drink == "Americano")
27   price = 1.20 + 0.30*(ounce/8);
28
29 price = price + (shots - 1)*.50;
30 price = price + price*taxRate;
31
32 /*
33 3.1008
34 */
```

Figure 17.3 The Espresso Pricing Program run using Scratchpad.

Summary

- In basic programming, you now understand the following:
  - Name–value separation is an important concept. It's one of the ways that programming differs from algebra
  - Letter sequences that make up a variable's name (identifiers) must be declared
    - Variables can be initialized when they are declared
    - Changing the value of a variable is possible by using assignment

Summary

- An assignment statement has a variable on the left side of the symbol and an expression on the right side
  - The operation is to compute the value of the expression and make the result the new value of the variable
  - This makes information flow from right to left in an assignment statement
  - Statements like x = x + 1; make sense in programming, but not in algebra

Summary

- There are three JavaScript data types of interest to us—numbers, strings, and Booleans—and we can build expressions to compute values of these types
- Standard arithmetic operators and relationals compute on numbers, and logical operations on Booleans
  - In defining concatenation, you learned about “operator overload”
  - Expressions “do the computing” in programs and are generally a familiar idea

Summary

- As a rule, all programming statements are executed one after another, starting at the beginning
  - The conditional statements are the exception. JavaScript's two conditional forms are if and if/else
  - These allow statements to be executed depending on the outcome of a Boolean expression called a predicate

Summary

- We must be careful to group statements within a compound statement to make it clear which statements are skipped or executed
  - We also must be careful when using if/else in a conditional so that the if and else associate correctly

Summary

- The espresso program illustrates most of the ideas in this chapter
  - The program uses both numeric and string data types, as well as the declaration, assignment, and conditional statement forms
- All that keeps us from running the program and demonstrating our knowledge is setting up the input to acquire the values for drink, ounce, and shots, and outputting the price. This requires a UI written in HTML (the topic of Chapter 18)