Lesson 7: If Statement and Comparison Operators

OBJECTIVES: In this lesson you will learn about

- Branching or conditional satements
- How to use the comparison operators: ==, !=, < <=, >, >=
- How to use if and if ... else to evaluate conditions and make decisions

Preparing to Program

It is often useful to take a course of action depending on some circumstance. It does not always rain, so you do not always take an umbrella with you. But if it is raining, or is expected to rain soon, you would be wise to take your umbrella.

Programs and Web pages are often faced with a similar situation. For example, we have already discussed the fact that a Web page with JavaScript works differently depending on whether you are using Netscape or Internet Explorer. So it is useful for a Web page to determine if you are using Netscape or Internet Explorer, and to execute different code depending on the result.

In programming, code that asks a question and executes different paths depending on the answer is known as a **branching** or **conditional statement**. Picture in your mind a branch in the shape of a fork: you can go either left or right. The term conditional statement implies that execution depends on a condition. If the condition has a certain value, do this; if it does not do something else.

This lesson will teach you how to use the **if** and **if** .. **else statement**. The if and if ... else statements are examples of a conditional statement. Conditional statements work in the following way: they ask a question, then execute certain code depending on the answer. In JavaScript, and in most other programming languages, conditional statements ask a question by using **comparison operators**. Before we discuss the syntax of the if statement, we need to explore the topic of comparison operators.

Comparison operators

Comparison operators are used to make comparisons. For example you can compare two variables to test if they are equal. Other comparisons are available (see the table below). For each of the comparison operators, the result of the comparison is always either **true** or **false**. True or false values are known as **boolean values**.

| TT ' 11 11 1 1 1 | .1 | . 111 | · T O · / |
|--------------------------------|----------------------|-----------------------|---------------------|
| Here is a table that describ | s the comparison one | rators available to v | Jou in Java Scrint |
| Tiere is a table that describe | s the companison ope | rators available to | you iii savasciipi. |

| | Meaning | Comments |
|----------|--------------------------|--|
| Operator | | |
| == | equal | True if the two operands are equal; false otherwise. |
| != | not equal | True if the operands are not equal; false otherwise. |
| < | less than | True if the left operand is less than the right operand; false otherwise. |
| <= | less than or equal to | True if the left operand is less than or equal to the right operand; false otherwise. |
| > | greater than | True if the left operand is greater than the right operand; false otherwise. |
| >= | greater than or equal to | True if the left operand is greater than or equal to the right operand; false otherwise. |

These operators should already be familiar to you. You have used them in math class to make comparisons.

Here are some simple examples that demonstrate how to use comparison operators:

Assume: a = 7, b = 4 and c = 10. a < b is false. a < b + c is true. a * b >= 2 * c is true

Assume: name = "Pace",
state = "NY",
address = "1 Pace Plaza"

name <= state is false

name > address is true

a+b+c == 21 is true state == address is false

a+b != 0 is true

The if Statement

The if statement is an example of a conditional or branching statement. The if statement works in the following way: it asks a question, normally by using comparison operators. Depending on the answer, it will execute certain code.

Here is the general syntax for the if statement:

```
if ( condition )
{
    JavaScript statements go here
}
```

If the *condition* is true, the *statements* between { ... } will be executed. If the *condition* is false, the *statements* between { ... } will be skipped. The group of statements between { ... } is called the **if block**. Any statements may be placed in this if block, including additional if statements.

Recall that you used curly braces { and } to define the boundaries for a function in the last lesson. Notice also, that the *condition* can only be true or false. In programming, you can only ask yes/no or true/false questions. If you ever played the game 20 Questions, you know it can be quite a challenge to figure out something by only using yes/no questions. This is one of the restrictions that can make programming a difficult task.

Example

```
if (city == "New York")
{
   state = "NY"
   areacode = "212"
   document.write(city, state)
}
```

If the condition **city == "New York"** is true, all of the statements in the if block will be executed. If the city does not match "New York," none of the if block's statements will be executed.

The if ... else Statement

An **if** statement can also have an **else** clause that executes if the condtion is false. This enables a two-way branch so that one block of statements can be executed when the condition is true, and a second block of statements can be executed if the condition is false.

Syntax for if ... else: if (condition)

```
statementgroup1
}
else
{
statementgroup2
}
```

If the *condition* is true, *statementgroup1* (the if block) will be executed and *statementgroup2* will not be executed; otherwise (when the condition is false), *statementgroup1* will not be executed and *statementgroup2* (the **else block**) will be executed.

Example

```
if (city == "New York")
{
   state = "NY";
   areacode = "212";
}
else
{
   state = "CT";
   areacode = "203";
}
   document.write(city,state);
```

In this example, if city has the value "New York," then the two statements in the if block will be executed and the statements in the else block will not. If city does not have the value

"New York," the statements in the if block will be skipped and the statements in the else block will be executed. This is why the if ... else statement is called a branch.

Note

When the **if** block and/or the **else** block have only one statement, the curly braces can be omitted. If the braces are omitted, the statement *must appear* on the same line as the if and/or else. For example:

```
if (city == "New York") state = "NY"
else state = "CT"
document.write(city,state)
```

if Statement and Browser Detection

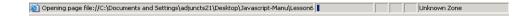
Recall at the beginning of the lesson we mentioned that it is useful for a Web page to determine what broswer is being used to view the page. This can be done with an if ... else statement. Here is what the code looks like:

```
var browserName = navigator.appName;
if (browserName == "Netscape")
{
    alert("You are using Netscape");
} else if (browserName == "Microsoft Internet Explorer")
{
    alert("You are using Internet Explorer");
}
```

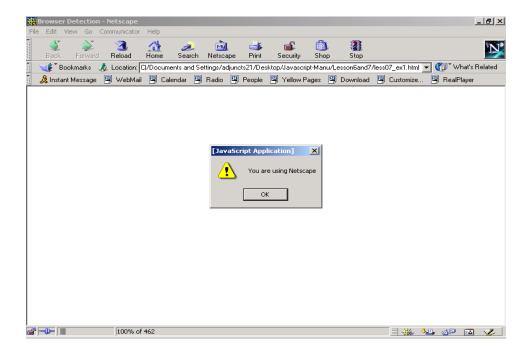
If you insert this code in a Web page then open it with Internet Explorer you will see the following message:







If you open it with Netscape, you will see this message:



What do you think will happen if you open this page with a browser other than Netscape or Internet Explorer?

This code works in the following way. The variable browserName is assigned a value from navigator.appName. The navigator object is another example of a JavaScript built in object. The navigator object stores information about what browser is being used. The appName property stores the name of the browser being used. Another navigator property available is appVersion, which indicates what version number of the broswer is being used.

In the if statement, browserName is compared for equality with the string "Netscape" by using the == operator. If the comparison is true, then the alert box announces you are using Netscape. If the comparison is false, it makes a second comparison. It now compares browserName with the string "Microsoft Internet Explorer." If they are equal, the alert box announces you are using Internet Explorer. Many Web pages use JavaScript to determine what browser you are using and then they execute different code depending on the result.

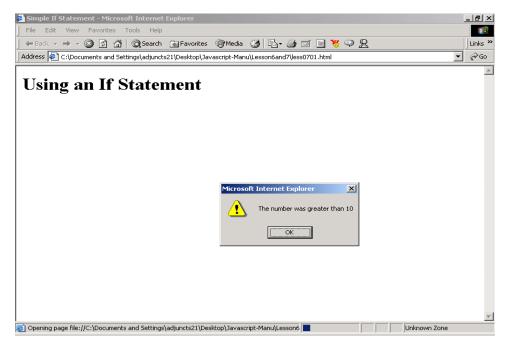
In the Lab

In this lesson you will learn how to use comparison operators and the if and if .. else statements.

Start 1st Page 2000 and begin a new HTML document. Save it giving it the name lesson0701.html.

Now type in *exactly* the following code:

If you run this code and enter the number 11, you see this result:



This code prompts the user to enter a number. The number entered by the user is stored in the variable number in numeric format by the JavaScript function parseInt. Refer to Lesson 04 for a description of how parseInt works.

Next the code performs a simple if statement. The number entered is compared with 10. If it is greater than 10, then and only then is an alert box displayed. Run the program a few times with numbers above 10 and below 10. If they are above 10, your code should display an alert box. If the number entered is below 10, it should do nothing. What happens when you enter 10 exactly?

Student Modifications

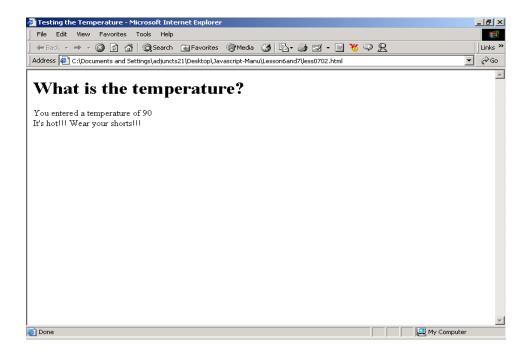
- add an else clause to the if statement that displays an alert box with the message that the number was less than or equal to 10.
- instead of comparing the input number to 10, prompt the user to enter two numbers and compare the first number to the second number, i.e. display either "The first number is greater than the second number," or "The first number is less than or equal to the second number," depending on the result of your comparison.

Save your code from the previous exercise and start a new document. Save it using the name lesson0702.html.

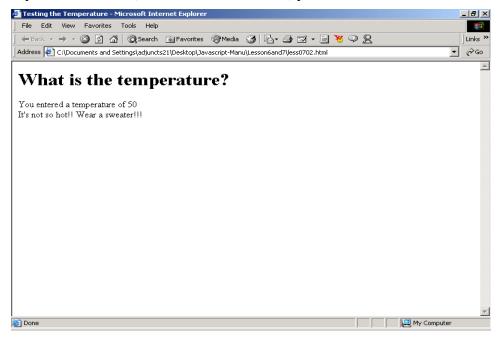
Now type in *exactly* the following code:

```
<html>
<head>
   <title>Testing the Temperature</title>
<body>
<h1>What is the temperature?</h1>
<script language="Javascript">
<!--
var temperature = parseFloat(prompt("Enter the outside temperature in
Fahrenheit", " "));
document.write("You entered a temperature of ",temperature,"<br>");
if (temperature >=80)
 document.write("It's hot!!! Wear your shorts!!!<br>");
else
   document.write("It's not so hot!! Wear a sweater!!!<br>");
//-->
</script>
</body>
</html>
```

If you run this code and enter a value of 90, you will see this result:



If you run this code, and enter a value of 50, you will see this result:



This code uses an if statement to test the value of the temperature. It prompts the user to enter a number, and converts the number to floating point format using the JavaScript function parseFloat. Recall from Lesson 04 that parseFloat stores numbers that can have a fractional value. Try this code a few times, entering both whole numbers and numbers with a fractional value. Enter the number 79.999. What is the result? Enter the number 80.001. What is the result?

Student Modifications

- Most of the world uses Celsius or Centigrade temperature values. What is considered a hot temperature in Centigrade? Re-write this program to ask the user to enter a Centigrade temperature. You also have to change the if statement to reflect a Centigrade value.
- Add additional if ... else statements to test different levels of temperature. Here is what your if ... else statements should look like:

```
if (temperature >= 100)

{
    alert ("wow is it hot!")
}
    else if (temperature >=90)

{
    alert (.....)
}
    else if ....
```

JavaScript Guessing Game

Now you will use the if statement in a guessing game. This program prompts the user to guess a number between 1 and 10. If the guess is correct, an alert box tells the user they guessed the right number. If it is not correct, the alert box tells the user they guessed the wrong number.

Save your code from the previous exercise and start a new Web document. Save it using the name lesson0703.html.

Now type in *exactly* the follwing code:

```
<html>
<head>
  <title>JavaScript Guessing Game</title>
   <script language="Javascript">
<!--
function makeGuess()
{ var guess = parseInt(prompt('Guess a number from 1 to 10', ' '));
   if(guess == number)
      alert('You guessed right :)')
   else
      alert('Wrong number guess again :(');
}
//-->
</script>
</head>
<body>
<center>
<h1>Guess The Right Number</h1>
<h2>Welcome to the guessing game!</h2>
<h2>Guess a number between 1 and 10</h2>
</center>
<script language="Javascript">
<!--
var number=(Math.floor(Math.random()*10))+1;
```

```
//-->
</script>
<center>
<input type="button" value="Make a Guess" onClick="makeGuess();">
</center>
</body>
</html>
```

This code is the start of a guessing game. It allows the user to make a guess. If you run this code and guess the wrong number, you get the following message:



If you run this code and guess the correct number, you get this message:



This code works in the following way.

The computer has to first "pick" the secret number for you to guess. It does this in the following line:

```
var number=(Math.floor(Math.random()*10))+1;
```

Although this code looks quite complicated, it is basically doing a simple thing – guessing a number from 1 to 10. In order to do this, it uses two JavaScript functions, **Math.random()** and **Math.floor()**. Math.random() is a special function that generates random numbers. Random numbers are very useful in computer science. They are used by many computer games, and can be used to simulate animation, and also to generate passwords.

Math.random() generates a random fractional number between the range of just above 0 (zero) up to 1. This code takes the fractional number from Math.random(), multiplies it by 10 and adds 1. This now expands the range of the random number from just above 1 (i.e. 1.413 or 1.232) to just above 10 (i.e. 10.003, or 10.4411.

For our guessing game, we are not at all interested in any of those fractional values; we just want to chop them off. That is what Math.floor() does for us; it chops off any fractional numbers, leaving just the whole number, from 1 to 10. Once we have this number, the assignment operator (=) stores it in the variable number.

After the computer has "picked" the secret number, the user makes a guess by clicking a button. The onClick event handler for that button now executes the makeGuess() function. Recall in the last lesson we learned that functions are often used with event handlers to respond to user events once the code becomes more than one or two lines. The makeGuess() function prompts the user to enter a number. It uses parseInt to store the user's number in numeric format in the variable guess. It then uses an if statement to compare guess with number. If they are equal, the user has guessed correctly; otherwise they have not guessed correctly.

This is just the start of the Guessing Game. Now we are going to start making some additions. After a few wrong guesses, you may get tired of the game and want to see what the right answer is. We will now add a button and a function to let the user see what the correct number is.

Add another button to this code in the line right after the first button by entering this code on a new line:

```
<input type="button" value="I Give Up!" onClick="showAnswer();">
```

Now you need to add the showAnswer() function to your code. Place this in the head section right after the closing brace of the makeGuess() function, but before the closing script tag (i.e. after line 19. Here is the code you should insert:

```
function showAnswer()
{
  alert("The correct number is " + number);
}
```

If you run this code and click the "I give up!" button you will see the following display:



Student Modifications

- Provide a hint to the user if they guess the wrong number. Tell them if their guess was too high or too low. Use an if statement inside the makeGuess() function in order to do this. Be careful. Only give this hint if they guess the wrong number.
- Add a button with the value= "New Game" that changes the secret number. Add a new function rePlay() that re-sets the value of number. Place this code inside your function to reset the number

```
number=(Math.floor(Math.random()*10))+1;
```

Also display an alert message to the user to let them know they are starting a new game with a new number.

Key Terms and Definitions

- **branching or conditional statement** a statement in a programming language that asks a question or determines a condition and executes different instructions depending on the answer.
- **if and if ... else statement** if and if ... else are examples of conditional statements that are part of JavaScript. Both if and if ... else examine a condition. For the if statement, code is executed if the condition is true. For the if ... else statements, one block of code is executed if the condition is true and a separate block of code is executed if the condition is false.
- **comparison operators** operators that allow code to compare two values, and return a result of either true or false. The comparison operators available are == (equal), ! = (not equal), > (greater), >= (greater equal), < (less), and <= (less equal).
- **boolean values** these are values that can have only two possible states: true or false.
- **if block** section of code that is executed within an if statement if the tested condition is true.
- **else block** section of code that is executed within an if ... else statement if the tested condtion is false.
- **Math.random()** JavaScript function that generates a random number.
- **Math.floor()** JavaScript function that removes the fractional portion of a floating point number.

Lesson 7 Summary

The lesson taught you how to use conditional or branching statements. Conditional statements are a very important component of programming languages. Conditional statements ask a question, then execute different code depending on the answer. You learned that you can use the comparison operators to ask questions in JavaScript. Comparison operators compare two values, and return an answer of either true or false. You also learned that you can use the if and if ... else statements to branch to different code blocks depending on the result of a comparison. Finally, in the lab you used the if statement and comparison operators to create a simple guessing game.

Lesson 7 Exercises

7_1. Write a program that inputs the number of items to be purchased and inputs the price per item, then prints the discounted price. The purchase is discounted by 15% only if there are 10 or more items purchased.

Prompt the user to input the number of items purchased and store it in a variable named numItems. Prompt the user to enter the price per item and store it in a variable price. Then use an if statement to determine if numItems is greater or equal to 10. If this is true, discount the price by 15%. If total is a variable that contains the total cost of all the items, the code that applies the discount looks like this:

```
total = total * .85;
```

Display the total cost in an alert box. Include in the alert message to total cost and whether a discount was applied.

- 7_2. Modify the previous program to allow a second level of discount of 25% for 50 items or more. Add another variable called discount, and set its value to .25 if they have purchased 50 or more items, .15 if they have purchased at least 10 to 49 items, and to 0 if they have purchased less than 10 items. Use the value in discount to calculate the final total and display it using an alert box. Include in the alert box the total amount due and the amount of the discount (if applicable).
- 7_3. Using the previous programs as guides, create a page with JavaScript that asks in two prompt boxes for the quantity and the price, and then calculates and displays the total price including:
 - no discount for totals below \$100
 - a 15% discount only if the total price exceeds \$100.
 - a 25% discount only if the total purchase exceeds \$1000.
 - a 30% discount only if the total purchase exceeds \$10,000.
- 7_4. Write a JavaScript program that prompts the user to enter their final average as a number. Use an if ... else statement to display their grade as a letter equivalent, i.e. 90 or above is an A, 80 or above is a B, etc.