Lesson 6: Introduction to Functions

OBJECTIVES: In this lesson you will learn about

- Functions
- Why functions are useful
- How to declare a function
- How to use a function
- Why functions are used with event handlers

Preparing to Program

As programs become larger and more complex, they need to be structured and organized. This is true for most things. Think about how textbooks are organized. There are chapters, sections, and units that break up the material into smaller pieces. Smaller pieces are easier to understand and create.

The same idea applies to programs and programming languages. As programs grow in size, they need to be broken up into smaller pieces. These smaller parts are called functions, methods, sub-routines, or sub-programs, depending on the programming language. JavaScript uses the term **function** or **method** to describe these smaller units of code. There is no formal definition to describe when you use the term function or method. In general, methods are associated with objects, and functions are free standing, not connected to a particular object.

A function or method is a unit of code that performs a specific, well defined task. You have already used these JavaScript functions:

- document.write – to write text to a Web page
- alert– to display a message inside a small window with an OK button
- prompt– to collect input from the user and store it in a variable

There are many other JavaScript functions that you can use. These functions give additional power and utility to your code. Functions are collected into **function libraries**. Much of the effort of learning a programming language centers on becoming familiar with what function libraries are available for use.

Why Functions Are Useful

Besides giving some structure and organization to your code, functions make programming much easier. As you program more, you will find that you are often repeating and rewriting similar sets of instructions again and again. Using functions means you do not have to repeat your code. Instead of copying the code over and over again, you can place it inside a function and then use the function repeatedly.

In JavaScript, functions are often used as event handlers. Recall that when you write code for an event handler, you must insert the code inside the definition of the link, button, or other Web element that generates the event. If you write more than one or two statements, it gets very messy. The preferred technique is to place the statements inside a function, and to then execute the function when the event handler is triggered.

Defining Functions

This lesson will show you how to define and use your own functions. The code that defines your function is normally placed in the head section of your HTML document. This allows your function to be loaded with the page and available for use when needed.

You begin by first declaring your function. A **function declaration** is similar to a variable declaration. In the first line of your function, your declare it as a function, give it a name, and indicate if it accepts any **parameters**. To declare a function, you start with the key word *function,* followed by its name, and then a set of parentheses:

```
function functionname()
```

The same rules for variable names also apply to function names. Function names must begin with a letter or underscore (_), and can only contain letters, numbers, and underscore. Function names, like variable names, are case sensitive.

We will briefly defer the discussion of parameters. After the first line, there is a ***required*** open curly brace {. After the opening brace, you place all the code to be executed when the function performs its task. The end of the function is marked with a closing curly brace }.

The general structure of a function is

```
function function-name ()
{
  JavaScript statements
  go here
} // end of function-name
```

The left brace defines the beginning of the function body, and the right brace indicates the end of the function body.

Parameters and Functions

Functions carry out tasks. Often, it is useful to have data available for the function to use when performing its task. Data is provided to a function by using parameters. You have already used parameters with JavaScript methods like alert and document.write. When you use these methods, you provide information within the parentheses that the function uses to carry out its task.

The syntax for a function with parameters is as follows:

```
function functionname(parametername1, parametername2,...)
```

These parameters are special variables made available to the function. Each parameter name must be a legal variable name, and that name must be unique within the list of parameters. Notice that each name is separated by commas. A function can have zero or more parameters. Even if a function has no parameters, the parentheses after the function name is required.

Return Statements and Functions

A common task that functions perform is some kind of calculation. For example, you could write a function that calculates the sales tax on a purchase. Once the function determines its result, it returns the result back to the main part of your program. This is accomplished by using the keyword **return** statement. The syntax of the return statement is straightforward:

```
return variablename;
```

For example, to return the calculated sales tax amount, you include the following code in your sales tax function:

```
return salesTaxAmount;
```

Calling Functions in JavaScript

There are two code sctions required when you use functions. The first part, described above, declares the function. It has specific syntax requirements, and it usually placed in the head section of your Web document.

The second part is the code that actually allows you to use the function. In programming languages, using a function is referred to as a **function call**. A function is not executed until it is called. It is available on standby for use, but will not be executed unless a statement inside your page calls that function.

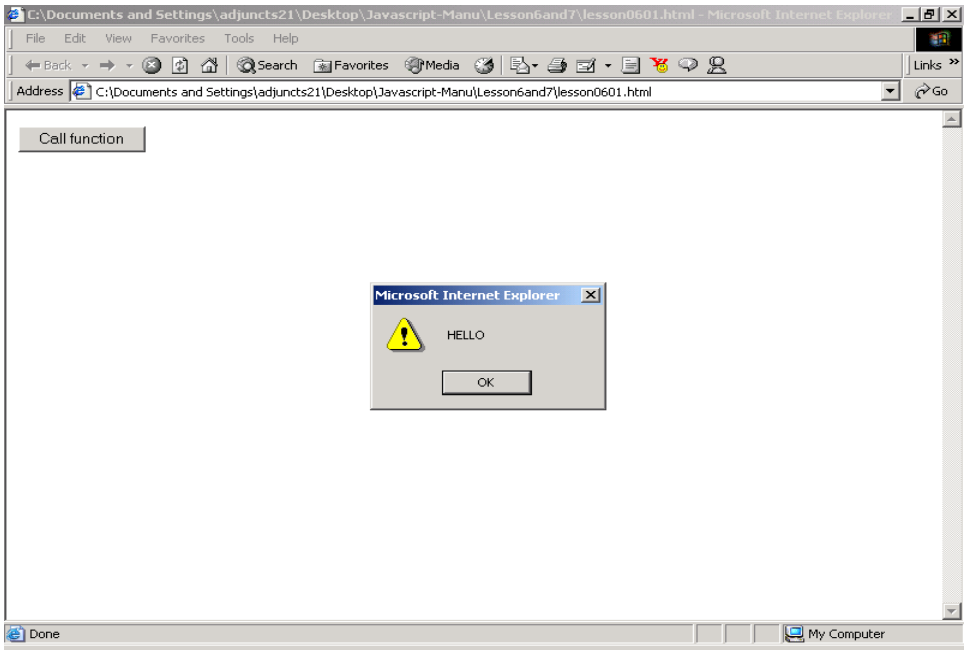
The syntax for a function call is also straightforward. If the function has no parameters, than the function call looks this:

```
functionname();
```

For example, here is a small JavaScript program that calls a function without parameters.

```
<html>

<head>

<script language="javascript">

function myFunction()
{
alert("HELLO");
}
</script>

</head>

<body>

<form>

<input type="button"

onClick="myFunction();"

value="Call function">

</form>

</body>

</html>
```

When executed, this code displays the following:



The function is named myFunction. It is declared within the head section of the Web document. Notice that you still must use the script tags for the function declaration. The function call is within the event handler onClick. Once you click on the button, the event handler calls the function. myFunction executes, and it displays an alert message. Although this is a simple example, it demonstrates the basic syntax of function declarations and function calls.

Calling a Function With Parameters

When calling a function that has parameters, the function call must include the exact number of parameters as required by the function. If the function has two parameters, the function call must include two parameters.

A function call with parameters takes the following form:

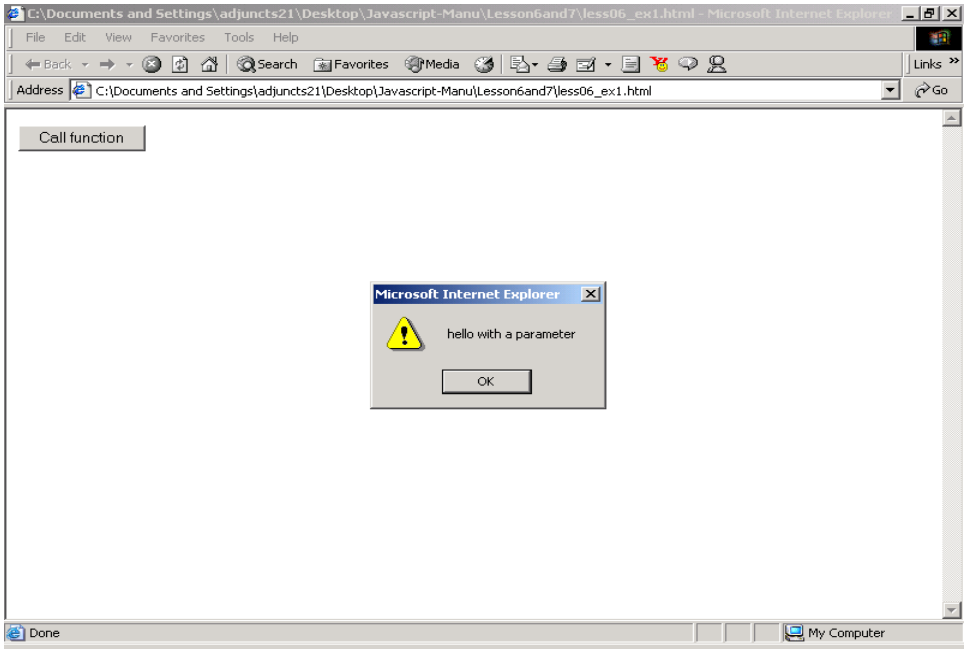
```
functionname(parametername1, parametername2, ...);
```

The code above can be modified to add a parameter to myFunction. Here is what the revised program looks like:

```
<html>

<head>

<script language="javascript">

function myFunction(message)
{
alert(message);
}

</script>

</head>

<body>

<form>

<input type="button"

onClick="myFunction('hello with a parameter');"

value="Call function">

</form>

</body>

</html>
```

Notice the following changes. myFunction now has a parameter called message inside its parentheses. The alert statement uses the parameter message when it produces its message box. And the function call now has a value inside its parentheses; that is passed to the function.

When this code executes, the following is displayed:



With this version, myFunction is defined with one parameter. The value of that parameter is used by the alert statement. The function call is once again triggered by the onClick event handler. The words 'hello with a parameter' are now passed along to the function when it is called by the event handler. If you change the words inside the function call, the result of the function will be different.

Function Calls and Event Handlers

Functions are often used in conjunction with event handlers to respond to user events. One simple reason is that event handler code looks cleaner and is easier to understand if you use a function. To demonstrate this, consider this example. The following code does not use a function to handle an event.

```
<html>

<head>
   <title>Event Handler Without Function</title>
</head>

<body>

<form>

<input type="button"

onClick="alert('one');alert('two');alert('three');alert('four');"

value="Alerts Without Function">

</form>
```

```
</body>

</html>
```

This crams all the statements between a set of quotes next to the event handler. If you make a mistake, it can be difficult to spot. If you wanted to add anything, you could start to have problems. The solution is to use a function.

Here is the same program re-written using a function:

```
<html>

<head>
   <title>Event Handler Without Function</title>

   <script language="Javascript">

<!--
function someAlerts()
{
  alert('one');
  alert('two');
  alert('three');
  alert('four');
}
//-->
</script>

</head>

<body>

<form>


<input type="button"

onClick="someAlerts();"

value="Alerts With A Function">

</form>

</body>

</html>
```

Instead of cramming many statements on one line, the event handler has a single statement, the function call itself. When the event is triggered, the event handler calls the function, and the function displays the four alert statements. In the lab you will use functions that will be used by event handlers to respond to user events.

In the Lab

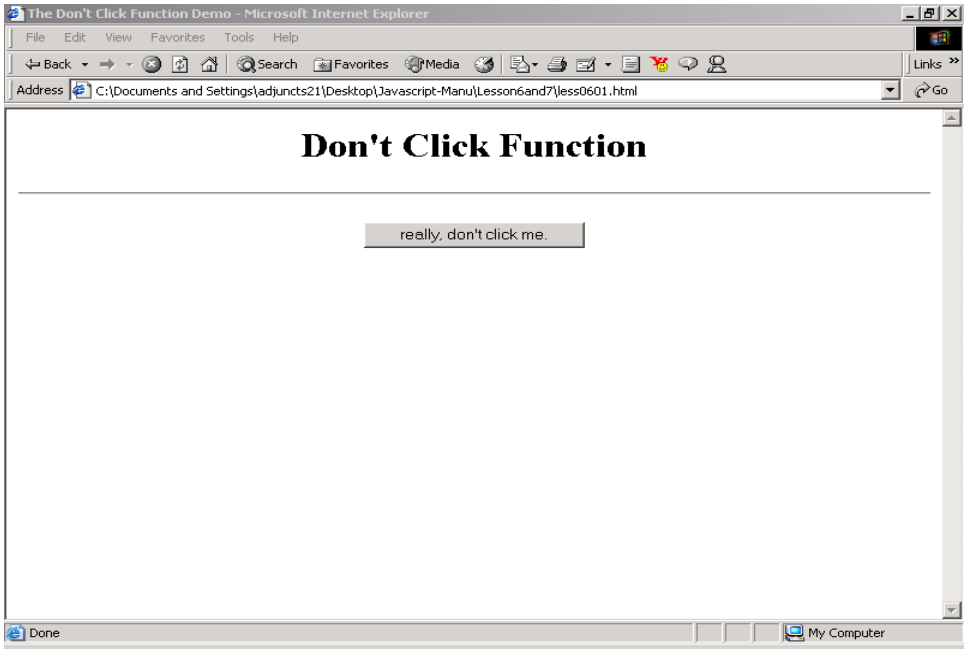
This week in lab you will use learn to write your own functions and use them in your code.

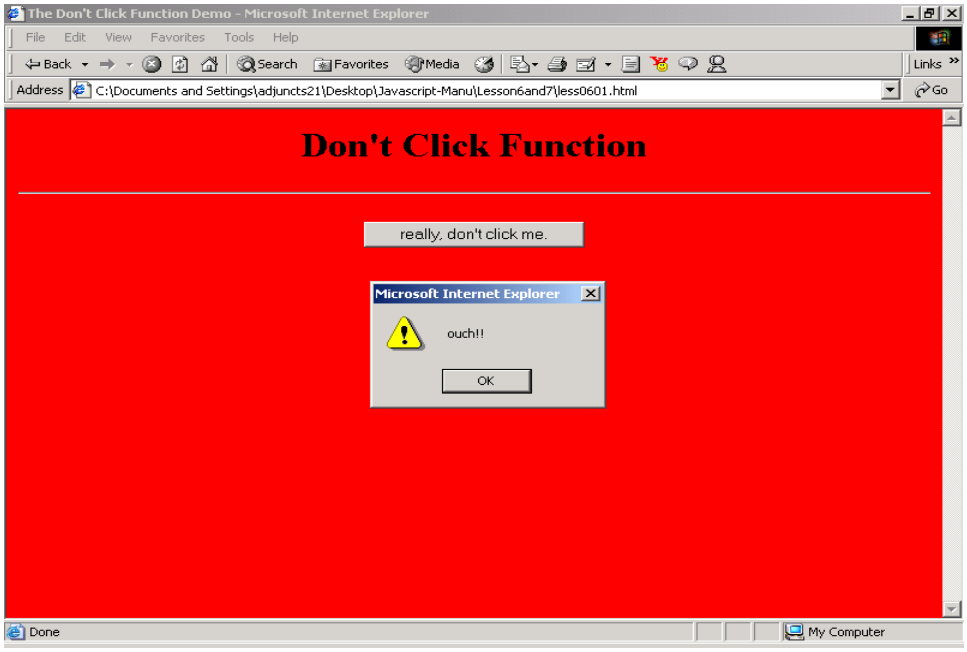
Start 1st Page 2000 and begin a new HTML document. Save it giving it the name lesson0601.html.

Now type in ***exactly*** the following code:

```
<html>

<head>

<title>The Don't Click Function Demo</title>

<script language="Javascript">

<!--

    function sayOuch()

    {

      document.bgColor = "red";

      alert ("ouch!!");

      document.bgColor = "white";

   } // end of function

//-->

</script>

</head>

<body>

<center><h1>Don't Click Function</h1>

</center>
<hr>

<center>

<form name = myForm>

<input type = button

   value = "really, don't click me."

   onClick = "sayOuch();">

</form>

</center>

</body>

</html>
```

If you enter the above code correctly and run it, you will see the following output:





Notice that you place the function declaration in the head section of your html document, inside the same script tags you have already used. Notice also the function call from the onClick event handler. When the user clicks the button, the function sayOuch() is executed. The function sayOuch() changes document.bgColor to red, displays an alert box with the word ‘ouch!’, and then changes the document.bgColor back to white.

Student Modifications

- change the colors used by the function.
- add a second button and second function that use different colors and a different message. Test it out using both buttons.

Using a Function With a Parameter

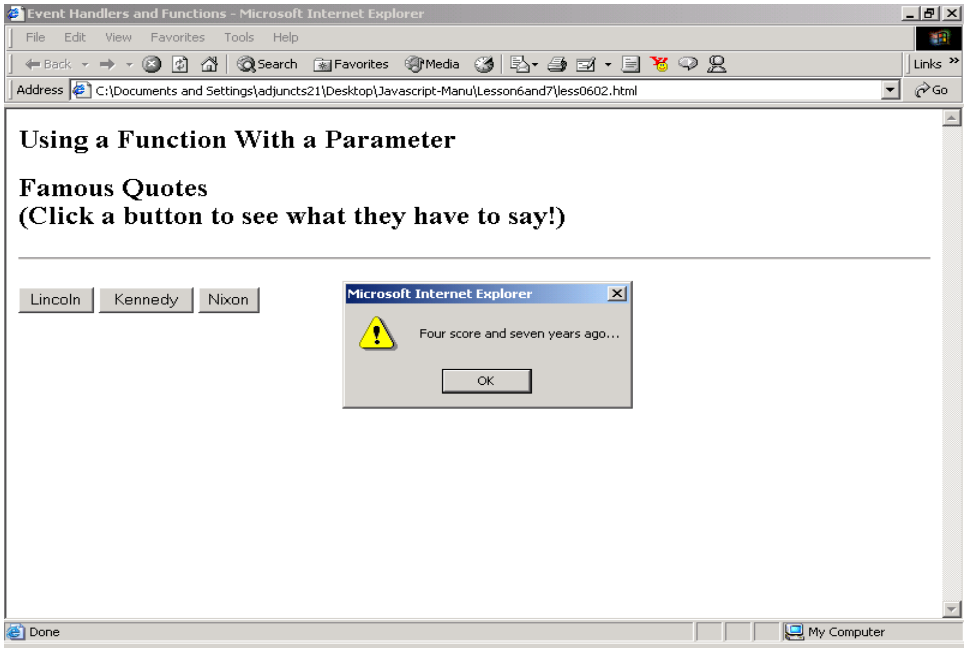
Save your work from the previous exercise. Start a new document, and name it lessson0602.html. This example will show you how to use a function with a parameter.

Now type in ***exactly*** the following code:

```
<html>

<head>

   <title>Event Handlers and Functions</title>

   <script language = "Javascript">

   <!-- Hide script from older browsers

    function saySomething (message) {

       alert (message)

         }

    //end hiding script from older browsers -->

    </script>

</head>

<body bgcolor=white>

<h2>Using a Function With a Parameter</h2>

<h2>Famous Quotes <br>(Click a button to see what they have to say!)
</h2>

<hr>

<form>

<input type="button" value="Lincoln"

onClick="saySomething ('Four score and seven years ago...')">

<input type="button" value="Kennedy"

onClick="saySomething ('Ask not what your country can do for you...')">

<input type="button" value="Nixon"

onClick="saySomething ('I am not a crook!')">
```

```
</form>

</body>

</html>
```

If you click the Lincoln button, the following is displayed:



This example also uses a function to display a message. Notice the same function can be used for all three buttons. By using a parameter, the message displayed can change.

Student Modifications

- add another button for another famous quote. Use the existing function, and display the message by using the parameter.
- add a second parameter, and use it to display a second alert box with the name of the person being quoted. You need to add the second parameter into the function declaration. Then you have to add another alert statement in the function, and change ***all*** the function calls to include the second parameter.

Key Terms and Definitions

- **function or method** – a set of statements that carries out a specific task.
- **function libraries** – collections of functions available for programmers to use in their programs.
- **function declaration** – required syntax that defines how the function carries out its task.
- **parameters** – special variables that are declared within parentheses along with the function. Parameters contain data the function uses in carrying out its task.
- **return statement** – return is a JavaScript key word that allows a function to send or return a value to the main program.
- **function call** – a statement that uses the function. Code inside functions does not execute without a function call.

Lesson 6 Summary

You have learned that when programs grow in size, it is important to organize and structure them by breaking them into smaller pieces. In JavaScript, these smaller pieces are called functions or methods. A function is a set of statements that carries out a specific task. You can create your own functions by first writing a function declaration in the head of your Web document. The function declaration gives the function a name, lists its parameters, and inside a set of curly braces contains the statements that will execute when the function carries out its task. In order to use the function, you write a function call. The function call consists of the function name plus any required parameters. Finally, you learned that functions are often used with event handlers to organize and simplify a Web page's response to a user event.

Lesson 6 Exercises

6_1. Create a Web page that does the following. Write a function combineWords(word1,word2) that displays both words (i.e. the words in the parameters) in an alert box with a space between them. Be sure to place the function declaration in the head section of your document, inside script tags.

Then in the body section, add script tags and include JavaScript code that prompts the user to input two words. Store this input in two variables, word1 and word2. Then write a function call to combineWords, passing the value's input to the functions. Hint: use concatenation inside the function to combine the words.

6_2. Here is the code for less0501.html from Lesson 5:

```
<html>

<head>
   <title>Lesson 05: Introduction to Events</title>
</head>

<body>

<center>
<h1>Click your favorite color:</h1>

<a href="#" onClick="document.bgColor='red';">Red</a>

<p>

<a href="#" onClick="document.bgColor='blue';">Blue</a>

<p>

<a href="#" onClick="document.bgColor='green';">Green</a>

<p>

</center>

</body>

</html>
```

Rewrite this code using functions for each event handler. Name your functions clickRed(), clickBlue(), and clickGreen(). Place each function in the head section of your Web page. Change the onClick event handler for each link so that it calls the appropriate function. Inside each function, include code to change document.fgColor to a different value. Also include a statement that uses window.status to display the name of the color that was linked.

6_3. Rewrite exercise 4_2 using a function milesPerGallon that calculates and displays the miles per gallon in an alert box.

Here is the origianl description for the exercise:

Write a JavaScript program that calculates miles per gallon. Use parseFloat and a prompt to ask the user to input the total number of miles driven and store it in numeric format. Use parseFloat and a prompt to ask the user to enter the number of gallons consumed. Calculate the miles per gallon with the following formula:

milesPerGallon = milesDriven/gallonsConsumed

Write a function milesPerGallon(milesDriven). It should take the value passed in the parameter milesDriven and use it to calculate the miles per gallon. In the function, use an alert statement to display the result.

Then in the body section of your Web page, prompt the user to enter the number of gallons consumed. Store the user's answer in the variable milesDriven. Then write a function call to milesPerGallon, passing the value in milesDriven to the function.