Coding in JavaScript – functions

A function contains code that will be executed by an event or by a call to the function.

You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file).

Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that a function is read/loaded by the browser before it is called, it could be wise to put functions in the <head> section.

How to Define a Function

Syntax

```
function functionname(var1,var2,...,varX)
{
  some code
}
```

The parameters var1, var2, etc. are variables or values passed into the function. The { and the } defines the start and end of the function.

Note: A function with no parameters must include the parentheses () after the function name.

Note: Do not forget about the importance of capitals in JavaScript! The word *function* must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

Invoking Functions

Defining a function doesn`t execute it. You have to call the function for it to do its work. You can invoke (or call) any function defined in the current page. You can also invoke functions defined by other named windows or frames.

You should invoke a function by name. The name needs to be followed by a set of parentheses that contain zero or more values to be passed to the function. The number of values being passed should be equal to the number of arguments the function has. The values are assigned to the arguments in the order in which they appear in the list.

JavaScript Function Example

Example

```
function displaymessage()
{
  println("Hello World!");
}

displaymessage();
```

The return Statement

The return statement is used to specify the value that is returned from the function. So, functions that are going to return a value must use the return statement. The example below returns the product of two numbers (a and b):

Example

```
function product(a,b)
{
  return a*b;
}
println(product(4,3));
```

The Lifetime of JavaScript Variables

If you declare a variable, using "var", within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.

If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

More Examples

A function can even be recursive, that is, it can call itself. For example, here is a function that computes factorials:

```
function factorial(n)
{
    if ((n == 0) || (n == 1))
      return 1;
    else {
      result = (n * factorial(n-1) )
     return result;
    }
}
```

You could then display the factorials of one through 10 as follows:

```
for (i = 0; i < 10; i++)
{
   println(“ “+i+" factorial is: "+factorial(i));
}
```

Some JavaScript Predefined Functions

This section discusses JavaScript predefined functions such as escape, unescape, eval, isNaN, tostring, alert, and others.

Conversion and Comparison

escape(string) - Encodes a string from ASCII into an ISO Latin-1 (ISO 8859-1) character set which is suitable for HTML processing. It is passed the string to be encoded, and returns the encoded string.

unescape - Converts an ISO8859-1 character set to ASCII.

If a string is "My phone # is 123-456-7890", after the escape function, it is "My%20phone%20%23%20is%20123-456-7890". After unescape, it is "My phone # is 123-456-7890".

eval()- Converts a string to integer or float value. It can also evaluate expressions included with a string. In the example, value1 becomes 62.

```
value1 = eval("124/2") ,
```

isNaN(value) - If the value passed is a not a number, the boolean value of true is returned, if it is a number, it returns false.

```
if (isNaN(string1))
{
  document.write("String1 is not a number\n");
}
else
{
  document.write(String1 is a number\n");
}
```

parseFloat() - Returns floating point numbers the same as the parseInt function, but looks for floating point qualified strings and returns their value as a float. The first line below returns a value of 123, and the second line returns 9.683.

```
value1 = parseFloat("123")
value2 = parseFloat("9.683")
```

parseInt()- Converts a string to an integer returning the first integer encountered which is contained in the string. In the example, value1 becomes 12.

```
value1 = parseInt("12b13") ,
```

If no integer value were found such as in the string "abcd", then a value of 0 is returned.

Methods that belong to all objects

toString() - Converts an object to a string. The syntax is shown below. Radix is the base value for the conversion, which may be 10 for decimal conversion. The value 1 is used for binary conversion, and 8 is used for octal conversion.

```
object.tostring(radix)
```

Dialog Boxes

alert(message) - Displays an alert box with a message defined by the string message. Example:

```
println("An error occurred!")
```

confirm(message) - When called, it will display the message and two boxes. One box is "OK" and the other is "Cancel". If OK is selected, a value of true is returned, otherwise a value of false is returned. An example:

```
if (confirm("Select OK to continue, Cancel to abort"))
{
  println("Continuing")
}
else
{
  println("Operation Canceled")
}
```

prompt(message) - Displays a box with the message passed to the function displayed. The user can then enter text in the prompt field, and choose OK or Cancel. If the user chooses Cancel, a NULL value is returned. If the user chooses OK, the string value entered in the field is returned. An example:

```
var response=prompt("Enter your name")
if (response != null)
{
  println(response)
}
```

Predefined Core JavaScript Functions

The predefined core JavaScript functions are listed below.

- decodeURI()
- decodeURIComponent()
- encodeURI()
- encodeURIComponent()
- escape()
- unescape()
- eval()
- isFinite()
- isNaN()
- Number()
- parseFloat()
- parseInt()
- toString()
- watch()
- unwatch()

With the exception of watch() and unwatch(), you have already learned enough to be able to make use of all these functions. The watch() and unwatch() functions work with things called "object properties" to help in debugging. (As object properties are quite an advanced topic, and we haven't yet covered objects).

All these built-in functions either check or modify data in some way or another. To find out more details about each of these functions, go to the following URL:

http://www.informit.com/articles/article.aspx?p=29797&seqNum=3

Mathematical Functions

JavaScript provides an extensive library of predefined mathematical functions, including square root (Math.sqrt) and maximum (Math.max). These functions all have names beginning with the prefix "Math.", signifying that they are part a library of mathematical routines. (Technically speaking, Math is the name of a JavaScript *object* that contains these functions -- we will discuss objects later.)

When calling a function in JavaScript, the function name is written first, followed by the arguments in parentheses. For example, the call Math.sqrt(25) would return the value 5, while Math.max(4, 7) would return 7. The arguments to a function can be any valid JavaScript values, including expressions and variables. Thus, the calls Math.sqrt(3*4 + 13) and Math.max(2+2, 42/6) would similarly return 5 and 7, respectively.

Other Math functions in JavaScript

Math.abs
Math.min
Math.floor
Math.ceil
Math.round
Math.pow
Math.random

For more details on JavaScript's built-in or core functions, including those for Math, refer to this URL:

http://www.tutorialspoint.com/javascript/javascript_builtin_functions.htm