**JavaScript Objects and Functions: Part Two**

Let's look at a REAL example of JavaScript objects in action. In this example, I am going to instanciate two Image objects, which are used to represent GIF/JPEG pictures for use on a web page. I am then going to set the src (source) properties of the Image objects to appropriate URLs; these src properties are identical to the SRC attribute of an IMG tag.

Example:

var picture0 = new Image();

var picture1 = new Image();

picture0.src = "capitalA.gif";

picture1.src = "capitalB.gif";

Now, let's say that I have an IMG tag named "fred" on my web page.

Example:

<img src="capitalD.gif" width="54" height="54" name="fred">

I could access the image, fred, on my web page using something called the "simplified document object model", which allows me to use dot syntax to point to an element on an HTML page.

Example (abbreviated):

window.document.fred

Example (abbreviated; looking at SRC attribute of fred):

window.document.fred.src

Example (full line of code):

window.document.fred.src = picture0.src;

window is the instance of the Window object which represents the current web browser window. document is the instance of the Document object which represents the current HTML page in the web browser window. fred is the name of the IMG tag on the HTML page. src is the SRC attribute of the fred IMG tag.

In the above example, I am setting the src property for fred equal to the src property of the picture0 Image object, thus changing fred's picture from a graphic of the letter D to a graphic of the letter A! I've just performed an image switch! Hooray!

Now, I'm going to switch fred's image to picture1, the graphic of the capital letter B:

window.document.fred.src = picture1.src;

Woowee! I've switched that image again! I'm programming in JavaScript, doing something useful! Wow!

Alright, so this is not ALL the code that you'll need to deal with image switching, but it outlines the basic principles. First, you instanciate image objects with src properties set to URLs for the images that you want. Then, you refer to the named IMG tag on the page using dot syntax and the simplified document object model, and reset the src property of the named IMG tag equal to the src property of the desired instance of the Image object.

Almost every web site in creation uses image switching to make rollover effects on graphical hyper-references! Be warned, however: all images involved in a switch MUST have the same dimensions, the same width and height!

**The SteeringWheel Object Revisited**

We need to talk more about the honk() method of the SteeringWheel object, to illustrate additional basic JavaScript programming principles.

Our computerized car horn is capable of doing more than merely honking; it can play tunes, as well! There are two tunes it can make, along with the generic honk sound: Beethoven's Fifth Symphony Theme, and Mary Had a Little Lamb.

Here are the string values representing these car horn tunes:

"5th" for Beethoven's Fifth Symphony (dah dah dah DAAAAHHHH!).  
"MHALL" for Mary Had a Little Lamb.  
"honk" for the generic honking sound.

Now, I need to pass these string values into the horn() method of the SteeringWheel object. To do this, I need to place the desired string BETWEEN the parentheses of the function call operator for the horn method in my JavaScript command. When you do this, you are said to be "passing a parameter" or "passing an argument" to the method.

Example:

theWheel.horn("5th");

In the above example, I am passing the string value "5th" as an ARGUMENT or PARAMETER (these words are often used interchangeably here) to the horn method of theWheel. This causes the horn in theWheel to play Beethoven's Fifth rather than the standard honking sound. Note: everything I say about parameters and arguments in this section may ALSO be applied to functions, since methods are really functions attached to an object!

Let's have the horn play Mary Had a Little Lamb.

Example:

theWheel.horn("MHALL");

Ah, what a sweet sound!

By passing different arguments to a method, I can cause that method to behave in different ways.

In JavaScript generally, you will pass ARGUMENTS or PARAMETERS for many different reasons and to many different effects. You can pass any data type you like as an argument to a method, whether it be number or string or boolean or whatever. Each built-in JavaScript method will require different sorts of arguments, and will perform different kinds of actions.

Example:

theWheel.horn(false);

I passed the boolean value, false, to the horn method. The way the horn method is written, a false value passed as an argument prevents the horn from honking. Many methods can accept boolean values of true or false in this way.

Many methods are programmed to accept more than one argument. If you wish to pass more than one argument to a method, you must separate each argument with a comma (and an optional space, if you wish).

Example:

theWheel.horn("5th", "MHALL");

In the above example, the horn method first plays Beethoven's Fifth, then plays Mary Had a Little Lamb.

Example:

theWheel.horn("honk", "5th", "MHALL", "honk");

In the above example, the horn method plays a sequence of sounds, starting with a honk, moving to Beethoven's Fifth, then Mary Had a Little Lamb, and ending with another honk. In JavaScript, methods may accept one or more arguments, or a changeable number of arguments, depending upon how that method is written.

Let's look again at our real world example, using the Image object.

Example:

var picture0 = new Image(54, 54);

It turns out that the Image object constructor can accept TWO arguments, both integers. The first argument represents the WIDTH of the image in pixels, while the second argument represents the HEIGHT of the image in pixels. These arguments initialize the width and height properties of the instance of the Image object in question.

Example:

var picture0 = new Image(54, 100);

var myWidth = picture0.width;

var myHeight = picture0.height;

The variable myWidth is now equal to 54, which is the value of the width property of picture0. The variable myHeight is now equal to 100, which is the value of the height property of picture0.

**One Last Note on Variables**

Variables are boxes for information. By stating a variable name in JavaScript code, you are passing or manipulating the information contained in that variable.

Remember this example from a previous section?

var louie = 10.25;

var josie = louie;

By stating louie's name in the second line of code above, I was able to copy or pass the information from louie into josie by using the "gets" operator. In the same way, I can pass information from a variable into a method by stating the name of the variable between the parentheses of the function call operator.

Example:

var myHornSound = "5th";

var theWheel = new SteeringWheel();

theWheel.horn(myHornSound);

I just love to hear Beethoven's Fifth Symphony!

Example:

var myHornSound0 = "5th";

var myHornSound1 = "MHALL";

var theWheel = new SteeringWheel();

theWheel.horn(myHornSound1, myHornSound0, "honk");

In the above example, I passed THREE arguments to the horn method of theWheel; two of the arguments were variable names, and one was an actual string with no variable involved. Some of my arguments can be variables, while others of my arguments can be direct data; this is perfectly legal.

The following example is NOT legal:

var myHornSound = "5th";

var theWheel = new SteeringWheel();

theWheel.horn("myHornSound");

Notice how I passed the STRING "myHornSound" to the horn method when I REALLY wanted to pass the variable named myHornSound?  
"myHornSound" is just a bunch of text characters, a string, while myHornSound is an actual call to the variable itself. The above code would break your JavaScript, since the horn method does not recognize the string, "myHornSound", as a legal argument value.

**Summary**

A VARIABLE may hold an INSTANCE of an OBJECT. Each object has PROPERTIES and METHODS. Properties are essentially variables attached to an object, while methods are essentially FUNCTIONS attached to an object. A FUNCTION is a program which performs some action. Whenever you wish to execute a function (or a method), you must state the name of the function, followed by the FUNCTION CALL OPERATOR (which is represented by opening and closing parentheses characters). Properties and methods of an object are accessed using DOT SYNTAX, which allows you to trace the path of relationships between parent and child objects. Note: as you might expect, then, properties and methods of an object are child elements of the object. When you execute a method or function, you may pass ARGUMENTS or PARAMETERS to that method or function. Arguments/parameters passed to a method/function can influence the behavior of that method/function in some fashion. Data may be passed as an argument to a method/function either directly (by stating the actual number, string, or boolean value), or by passing a variable containing the data (by stating the variable name).