**Functions: Declaring Arguments**

You have already passed arguments to a built-in JavaScript method, the alert() method of the window instance of the Window object.

Example:

alert("Howdy");

Example (conservative syntax):

window.alert("Howdy");

As mentioned earlier, methods are functions which are attached to objects. Functions, then, can receive arguments. Now that you know how to code a function, I can show you how to declare arguments for your functions.

Example:

<html>

<head>

<title>Sample JavaScript</title>

<script type="text/javascript">

/\* <![CDATA[ \*/

function doStuff() {

alert("Howdy");

}

/\*]]>\*/

</script>

</head>

<body>

<p><a href="#" onClick="doStuff('Boo');">Calling doStuff()</a></p>

</body>

</html>

In the above example, I have declared one function, doStuff(). In the BODY of my page, I have called doStuff() from the onClick event handler in the hyper-reference; I am passing doStuff one argument, the string "Boo".

Now, how would I access that argument inside of the doStuff() function, and how could I pass the argument's data to the alert() method call in doStuff()?

The easiest way to access an argument inside a function (and the only one we're going to explore in this introductory lecture) is to declare a local variable which will hold the argument data; this local variable MUST be declared within the parentheses characters following the name of the function, doStuff, in the function declaration. Note: You would NOT use the keyword, var, when declaring this type of local "argument" variable.

Example (abbreviated):

function doStuff(fred) {

// some code here...

}

By declaring the local variable, fred, within the parentheses of the function declaration in the manner shown above, I am creating a container to hold the argument value being passed into the function. Within the function, then, I could access the local variable, fred, normally. Whatever argument has been passed to this doStuff function would be accessed via fred.

Example (abbreviated):

function doStuff(fred) {

alert(fred);

}

Now, in the above example, what value does fred actually contain, and what string would display in the alert dialog box? The answer is, it depends on what value has been passed as an argument to the doStuff function.

Example:

<html>

<head>

<title>Sample JavaScript</title>

<script type="text/javascript">

/\* <![CDATA[ \*/

function doStuff(fred) {

alert(fred);

}

/\*]]>\*/

</script>

</head>

<body>

<p><a href="#" onClick="doStuff('Boo');">Calling doStuff()</a></p>

<p><a href="#" onClick="doStuff('Ouch');">Calling doStuff() Again</a></p>

</body>

</html>

Here is the above example [**displayed**](http://fog.ccsf.edu/~srubin/js4.html).

When the first link calls the doStuff function from its onClick event handler, it passes one argument to doStuff(), the string "Boo". That string value is passed into the local variable, fred, because fred has been declared as a local "argument" variable within the parentheses of the doStuff function declaration. The variable fred now contains the string value "Boo", which is passed to the alert() method in the above code; the alert() method then causes an alert dialog box to pop up which says "Boo".

The second link performs the same actions, only passing the string "Ouch" instead of "Boo".

As you can see, both hyper-references call the SAME function, but each hyper-reference passes that function a different argument value. The function itself always performs the same tasks, but its behavior is altered slightly depending on the value of the argument passed into the local variable, fred.

This is the seed of how you create generalized functions to perform variations on the same task! Without these sorts of local argument variables, a function would be forced to behave identically EVERY time you called it, just like the functions you saw in the previous sections. WITH these local "argument" variables, a function becomes much more flexible in the tasks it can perform.

Just as I can pass more than one argument to a method/function, I can create more than one local "argument" variable by declaring those variables within the parentheses of the function declaration. Each of the local variable names declared between the parentheses of the function declaration would be separated by a comma.

Example (abbreviated):

function doStuff(fred, george) {

alert(fred);

alert(george);

}

Example (in context):

<html>

<head>

<title>Sample JavaScript</title>

<script type="text/javascript">

/\* <![CDATA[ \*/

function doStuff(fred, george) {

alert(fred);

alert(george);

}

/\*]]>\*/

</script>

</head>

<body>

<p><a href="#"

onClick="doStuff('Boo', 'Scared You!');">Calling doStuff()</a></p>

<p><a href="#"

onClick="doStuff('Ouch', 'That Hurt!');">Calling doStuff() Again</a></p>

</body>

</html>

Here is the above example [displayed](http://fog.ccsf.edu/~srubin/js5.html).

Arguments are always passed into the local "argument" variables for a function in sequence from left to right. The first argument passed to the above function would be passed into the first local argument variable, fred, and the second argument passed to the above function would be passed into the second local argument variable, george. The local "argument" variable name is irrelevant to which argument gets passed into which variable; the arguments are always passed into the function in order, left to right.

Being able to declare local variables to hold arguments for a function is an extremely powerful tool. Obviously, we haven't time to explore all of the ramifications of this capability, but I'll try to show you something simple but practical to do with this power in the next section.