JavaScript and Arguments

Kimerlee Holcom

Full Sail University

January 28, 2015

Just as there are many languages in the world today, there are many languages in the computer world as well. Although these languages are different, there are many similarities between most of them. One of these similarities is the use of functions to reduce the amount of code that is to be repetitively coded over and over. Functions allow the developer to write the code one time and use the code as many times as needed, without having to sit at a desk for hours typing the same thing many times. One thing that comes along with functions is the ability to pass arguments into them to get specific results. Arguments are values. These values could be words, sentences, numbers, values of true or false, or even groups of data stored in arrays. Each language deals with them differently, but JavaScript is a very interesting language in the many different ways that they can be handled.

First of all, JavaScript can simplify the handling of the number of arguments for a developer. For example, if a function is passed too few arguments, then JavaScript will use the arguments passed as the first parameters, and when it gets to a parameter where there is no argument, then it says that value is undefined. However, if there are too many arguments to the number of parameters, it simply ignores the excess arguments and they do not exist according to the function (Flanagan, 2011). Sometimes it is necessary to make sure that the arguments passed into a function are of a certain quantity in order for a specific outcome to be calculated. If the ratio of arguments to parameters is paramount to the running of the function, a developer may use an if/else statement to check if the arguments length is equal to the length of parameters needed for said function (Flanagan, 2011).

Another way that arguments can be handled in a function is quite amazing. It seems, most of the time, students are taught that if a function is created, then the number of arguments must equal the number of parameters. This, however, is not the case. Just as stated above, Flanagan (2011) says that JavaScript can take care of this itself, and needs no interaction from the developer at all. However, there are times that a function can have no parameters and yet receive arguments to be used to create the desired result. Axel Rauschmayer explains that there is a special variable that is associated with JavaScript. This variable is called *arguments*. This variable acts like an array, but does not have all the functionality of an array. However, it does allow the developer to use the *.length( )* method to check the number of elements inside the arguments variable (2014). With this being said, one way of handling a function with variables not associated with parameters is to use a for loop to iterate through the *arguments* “array” and deal with each argument individually (Saternos, 2014).

Finally, one last thing that can be done with arguments is to make them completely optional. In *Speaking JavaScript* by Axel Rauschmayer (2014), he uses the following example:

Function pair(x, y){

x = x || 0;

y = y || 0;

return [x, y];

}

Axel Rauschmayer goes on to explain if the function is passed no arguments at all, the function will return a result of [0, 0]. However, if the function is passed just a single argument, such as the number 2, it would see that argument as the parameter x, and return a result of [2,0]. Lastly, if two arguments are passed in, such as 8 and 4, then the requirements are met, and a result of [8, 4] would be returned (2014).

Functions can be tricky to grasp, but if a developer can understand how one language handles arguments, then that knowledge can be used in future endeavors with other languages, and the learning process is just that much simpler.

References

Flanagan, D. (2011). *JavaScript: The definitive guide* [eBook]. Accession Number: 415518. Retrieved from eBook Collections (EBSCOhost).

Rauschmayer, A. (2014). *Speaking javascript* [eBook]. Accession Number: 708634. Retrieved from eBook Collection (EBSCOhost).

Saternos, C. (2014). *Client-server web apps with javascript and java* [eBook]. Accession Number: 752049. Retrieved from eBook Collection (EBSCOhost).