**More on Simple Functions**!

All of this should already be in your class notes.

1. What is a function?
2. When is the function executed?
3. Is it executed when the page loads?
4. How to set up a function?

The function needs to be with in the script element or in an external JavaScript file. The function needs to have loaded before it is called.

Here is the template for a function that has no parameters and is named *get\_the\_A*.

function get\_the\_A () {

}

We put the content of the function between the curly brackets.

1. Write a function that has **no parameters** that puts out an **alert box** that says *I got the A!*

function get\_the\_A () {

alert (“I got the A!”);

}

1. How do we call that function?

get\_the\_A();

If there are no parameters a function call is the name of the function followed by opening and closing parentheses.

1. Where does the **function call** go?

Anywhere you can put JavaScript.

1. Where can you place JavaScript?
2. Set up a function that has one parameter.

function silly (a) {

}

We named our function ***silly***. The rules for naming functions are the same as the rules for naming variables.

In the function ***silly*** above we used the variable ***a*** as the parameter. We could have used any valid variable name.

When we call the function, the value we send will map into the variable ***a***. The variable ***a is initialized to the value that was sent in the function call.*** So if the function call is ***silly(“animal”);*** then the variable ***a*** is initialized with the value ***animal***. (The parameter (variable) ***a*** is assigned the value ***animal****.*) If we call the function again, **silly(“Goose”);** this time the string ***Goose*** maps into ***a*** so the initial value of ***a*** is ***Goose***.

1. Add some content to the function.

function silly (a) {

alert(“Is the “ + a +”silly?”);

}

1. Now look at the two function calls listed in number **9** above. What is the output of those function calls.
2. So we have written a function with no parameters, and one with one parameter. What if we want to write a function with two parameters?

First set it up. Make up a name for the function, *concat*, and two variable names, *x and y*, for the parameters.

function concat (x, y) {

}

What is the function to do? Concatenate two values and display them in an alert box.

OK, I could do this:

function concat (x, y) {

a = x + y;

alert(a)

}

In the example above, we concatenated the values that were sent to x and y and used a local variable (local to this function) named ***a*** to save the result.

Or do could do this:

function concat (x, y) {

alert(x+y)

}

We concatenated the values that were sent to x and y within the alert. The result would be the same. There are many ways to do things. We do not think of exact the same steps to solve the problem.

What else could we have done?

function concat (x, y) {

x = x + y;

alert(x)

}

Here we concatenated the values of x and y and replaced the value of x with the result. (Assigned the concatenated value to the variable x)

When the function is called, strings (not numbers) need to be sent to the function so it will concatenate and not add.

Let’s write the function calls.

concat(“soft”,”ware”);

concat(“computer “,”program”)

concat(“The dog “,”ran away!”)