JAVASCRIPT PART TWO GUIDE

This outline is a work-in-progress, and may change in the future – medatech@medasf.org

Mission Economic Development Agency | Instructor Eduardo Garcia

JAVASCRIPT IF STATEMENTS

The JavaScript if statement is a commonly used statement to let the program decide on a path to take. Generally these can be simplified to yes or no questions through the use of booleans. The following shows the general syntax of an if statement:

```
if (10 == 10) {
     console.log(“Hello there!”);
} else {
     console.log(“Goodbye!”);
}
```

- ➔ **if** this keyword is must be used to signify that the following code is part of an if statement, the if keyword must be lowercase and must be at the beginning of the statement.
- ➔ **(10 == 10)** The parenthesis is the condition to test, if the comparison inside is considered true then it will be converted to a boolean of true, otherwise it will convert to false. The condition must be inside of the parenthesis and it does not need to end in a semi-colon.
- ➔ **{ console.log(“Hello there!) }** This section is the code block, the code that will run if the condition was true, otherwise the program will skip over this code block and ignore it.
- ➔ **else { console.log(“Goodbye!”); }** if the condition was false, we can signal a second code block to run instead! Think of the whole if statement as “If this is true, do this, otherwise do that”.

If you want to test for additional conditions within the same if statement, you can add an **else if** section, similar to the else section of an if statement. Check out Mozilla Developer Network on how to use **else if**.

JAVASCRIPT FUNCTIONS

The JavaScript Function is also one of the most used code structures available to us. The basic usage of a function is to group up a bunch of statements that you want to reuse later on. Take the following example:

```
function sayHelloThreeTimes() {
     console.log(“hello there!”);
     console.log(“hello there!”);
     console.log(“hello there!”);
}
```

- ➔ **function** Similar to the if keyword, this signifies that the following code is part of a function.
- ➔ **SayHelloThreeTimes()** This is the name of the function, if you want to “use” this function later, you call it like you would a variable but you also need to write the parenthesis after the name (both while defining it and calling it). Instead of holding values though, it runs the code block instead, for every time you call it. If I did the following:

  ```
  sayHelloThreeTimes();
  sayHelloThreeTimes();
  ```

  Then I would get a total of six “hello there!” in the console! Each function call says hello three times!
- ➔ **{ ... }** The code block to run whenever you call the name of the function.

Bonus: Functions can be more complex if needed. Have a search online on “JavaScript function parameters”, “JavaScript function arguments”, “JavaScript function returning values”, “JavaScript passing functions”, and if you are really crazy “JavaScript recursive functions”.

JAVASCRIPT LOOPS

Sometimes we need the computer to run things for us for a certain amount of times, or based on certain value. For this guide we will focus on one of three main JavaScript Loops, the for loop. The for loop is a shorthand for the while loop, but it is the most commonly used loop. If you are confused on the for loop, I recommend checking out the while loop first to understand loops in general. The syntax for the for loop is as follows:

```
for (var i = 0; i < 10; i = i + 1) {
     console.log(
}
```

- ➔ **for** Yet another JavaScript keyword, signifying that the following code is a for loop.
- ➔ **(var i = 0; i < 10; i = i + 1)** This section is a little tricky. There are three parts to this set of parenthesis. The first part is the iterator, or the variable that will “count” for us, the letter i is commonly used for counting, usually representing the word *iterator* or *integer*. The second part is

the condition to test. if the condition is true, the code block will run and then check the condition again, if it is false, then it will stop the loop. The third part is the increment, this is the section that modifies the iterator so that one day the condition becomes false. *If the iterator never changes, then we have an infinite loop, meaning it will run forever, and the computer will try to run it forever, either freezing or crashing the browser or computer!* Note that the increment section runs after the code block has executed, it's usually the last thing to run in a loop iteration.

➔ **{ ... }** The code block that runs everything the condition is true. Once the condition is false, it will no longer run the code block and end the loop.

If you take a look at the example for loop, you will notice that the variable i will hold a zero. The increment will add a one to the current value of i and then store it in the same variable. The loop will continue to run until the condition is false, so once the variable is holding ten, it is no longer true as ten is not less than ten, and it will stop the loop.

NOTES