Javascript Functions – Recitation 8

What is a **function**? A function is a block of code that will be executed when it is called.

Why use functions? Functions allow users to reuse code, so the same code doesn't have to be rewritten multiple times.
For example: Every time a student signs up for a course, we want to greet him/her. Since there could potentially be hundreds of signing up for courses, we don't want to have to rewrite that chunk of code hundreds of times. Let's just put the code in a function called greetStudent, and we'll just call the function whenever someone signs up!

How do you write a function?

```
<script>
        function greetStudent(studentName, courseNum){
                document.write("Welcome" + studentName + "to" + courseNum);
        }
</script>
```

Tips for writing functions:
-they begin with a letter
-they can use any mix of letters, numbers, and underscores
-avoid reserved words like 'if' or 'else' for function names
-give your functions meaningful names so someone else looking at your code can infer what the function does without reading the code inside it

Function **parameters** are the **names** listed in the function definition (eg. studentName, courseNum).
Function **arguments** are the actual **values** received by the function when it is invoked, so if we call greetName("Joyce", 170);, then "Joyce" is an argument.

Try to refresh your browser. Nothing happens because we haven't **called** the function.

Calling JavaScript Functions

The code inside the function will be executed when the function is called. The function can be called directly when an event occurs like when a user clicks a button, and it can be called internally by JavaScript code too. Keeping the code that we had above, let's continue with these examples.

METHOD 1: Calling your function in HTML!

Step 1: Let's make an HTML button.

```
<body>
    <button>Greet Student</button>
</body>
```

Refresh the page, and you should see that nothing happens when you click the button.

Step 2: We want to be able to call our function greetStudent() whenever we click the button. In the same button:

```
<body>
    <button onclick="greetStudent('Joyce','CS170')">Greet Student</button>
    <script>
        function greetStudent(studentName, courseNum){
        document.write("Welcome" + studentName + "to" + courseNum);
        }
    </script>
</body>
```

When we're calling our function greetStudent, we're **passing in two arguments**:

1) Joyce (which corresponds to studentName in the greetStudent function if you scroll up)
2) CS170 (which corresponds to courseNum above)

BE CAREFUL! When passing in arguments, **ORDER MATTERS**. If the function takes in arguments (studentName, courseNum) like it does above, you need to make sure that when you call the function, you consider the order of the arguments: greetStudent('Joyce','CS170')

METHOD 2: Calling your function with JavaScript!

```
<script>

    function greetStudent(studentName, courseNum){
        document.write("Welcome" + studentName + "to" + courseNum);
    }

    var student = "Joyce";
    var course = "CS170";
    greetStudent(student,course);

</script>
```

The function can be defined above or below your other code. It won't work until it's called anyways. We set the variable student and the variable course. When we do greetStudent(student,course), we're passing in two arguments, student, which holds

the value “Joyce,” and course, which holds the value “CS170.” Remember, **ORDER MATTERS!**

Scope: how “far” a variable from its declaration can be used

-Local scoping: variables declared in a function can only be used within that function (local variables)

-Global scoping: variables declared outside functions can be used throughout the program (global variables)

-Local variables: come into existence when a function begins, and when the function ends, they vanish.

-Global variables: around all the time; never disappear after they’re declared.

-For information to be saved from one function call to the next, use a global variable since local variables are deleted once the function ends.

JavaScript Built-in Functions

Instead of having to write these functions, someone already wrote them, so all you have to do is call them to use them. Here are a few examples, but feel free to search for others.

Date()

```
<script>
    alert(Date());
</script>
```

Math.max()

```
<script>
    var value = Math.max(10, 20, -1, 100);
    document.write(“The max number is: “ + value);
</script>
```

fontcolor()

```
<script>
    var str = new String(“HeLLoOoOo World!”);
    alert(str.fontcolor(“red”));
</script>
```