## Overview

In this activity, you will continue practicing with variables and ultimately create JavaScript functions that use those variables to create a message or a value.

Using variables is a key part of programming and making your code legible to others. Recall the syntax for creating a variable and giving that variable a value:

| var myFirstVariable = 'variableValue'; |
| --- |

## Tips

* Variables are reusable, so developers can reuse a value in their code without having to re-declare it every time.
* Just like a variable, a function is a reusable block of code. A function can return a value or modify an existing value. That value can then be reused in your code in another function and so on.
* JavaScript makes a distinction between different types of variables. The two variables shown below (*var stringTwo = ‘2’ ;* and *var numberTwo = 2;*) might seem similar enough, but they’re actually very different! The stringTwo variable is a **text string** with the value of '2', while the numberTwo variable is a **number** with a value of 2. This is a complicated subject that won’t be fully covered in class, but more reading is available [here](https://www.w3schools.com/js/js_variables.asp) and [here](https://www.w3schools.com/js/js_datatypes.asp).

| var stringTwo = '2';  var numberTwo = 2; |
| --- |

## Instructions

**Step 1: Add Numbers**

1. Open index.html.
   * The file already has a <script> tag that links to index.js in the js folder.
   * **Hint:** You can check if your JavaScript file is correctly linked and loaded by using console.log() like you did in the last activity.
2. Open index.js.
3. Create a function named addNumbers.
   * **Note:** A JavaScript function has the following syntax:

| function doSomething() {  // Code here  } |
| --- |

* + **Note:** A function has four parts to its syntax:
    1. The function keyword specifies that the following code is a function.
    2. The name of the function is doSomething.
    3. The parentheses contain the function’s parameters. This particular function doesn’t accept any parameters, so there are no inputs within the parentheses (). If you wish to read about passing parameters to a function, read [here](https://www.w3schools.com/js/js_function_parameters.asp). We will not be passing parameters to functions in this class because we give you a very fundamental understanding of how the language works.
    4. The curly braces {} contain the actual code that will be executed when the function runs. This is represented by the comment // Code here.

1. Within the addNumbers function, add two variables, two and three.
   * Set the value of two to the **number** 2.
   * Set the value of three to the **number** 3.
2. Next, create a variable named five and set its value equal to the **sum** of two and three.
   * ***Don’t use numerals.*** Use text.
   * Just like the math you’d do on a calculator or phone, JavaScript allows you to add, subtract, etc. numbers and variables on your keyboard to create new values:

| var x = 1 + 2;  var y = x + x; |
| --- |

1. Next, run console.log() on the variable five.
2. Lastly, you need to call your function, so that it will run the code within it.
   * You can run a function by simply typing out its name:

| doSomething(); |
| --- |

1. Save your changes, open index.html in your browser, and open the Console tab in your browser’s inspector. You should now see the words “5” print out in your console!

**Step 2: Perform String Concatenation**

1. Back in index.js, create a new function named stringConcat. Remember the syntax example at the top of this document. Reference this example if you can’t remember the syntax of how to create a function.
2. Declare two variables, taco and cat, in this function.
   * Set the value of taco to the string 'taco' and the value of cat to the string 'cat'.
3. Create a third variable named tacoCat and set its value to the combined string of taco and cat.
   * Just like adding numbers, you can combine strings in JavaScript with the **+** operator.

| var newString = 'new' + 'String'; |
| --- |

* + What is the value of newString in this case?
  + What if you want a space between your strings? Easy! Just add a string space!

| var blueHat = 'blue' + ' ' + 'hat'; |
| --- |

* + Be sure to add a space string between taco and cat, so your message is legible.

1. Next, use another built-in function, alert(), to send a notice to the user.
   * alert() has similar syntax to console.log(), except it will open a window with a message instead of discreetly outputting a message to the browser’s console.

| alert('string'); |
| --- |

* + alert() the variable tacoCat to the user.

1. Call the stringConcat function.
2. Save index.js.
3. Open index.html in your browser and view your message!