Functions - created using function keyword + name ex. Function myfunction() - these parentheses signify a function
Parameters - Not required, they are what you put inside the () of the function. This is so the function can execute the same code using different variable values
Invoking functions - When you define a function it will not automatically run. You have to invoke the function using the function name + (). You can invoke the function before or after you define the function
Return values - these are values that return when a function is run. For example, if a function adds two words together, then the return value will be the new string created.
Scope - Only variables declared within a function, or of a broader scope, can be accessed by that function. Variables declared in a separate function cannot be used by another function on the same level. It is best to avoid using global variables
Arrays - a list of numbers, strings, or other things (used to store more than one value at a time), created using []
Index number - You can access an array element by using the index number of it. In JavaScript, index values start at 0
Changing values - you change the value of a specific element in an array by setting the index number = to the value you want to change it to
.length - returns the length of an array
First element in an array - accessed by using array name[0]
Last element in an array - accessed by using array name [array name.length - 1]
toString() - returns a string of comma separated array values, arrayName.toString
join() - joins every element in an array, but lets you choose how to separate them by entering that value as the parameter, arrayName.join(“ “)
pop() - removes the last value in an array, returns the value that was popped out
push() - pushes a value to the end of an array, returns the new array length
shift() - similar to pop but removes the first element in an array, returns the value that was shifted out
unshift() - similar to push but adds to the beginning of the array, returns the new array length
delete() - can delete items in an array, but shouldn’t be used because it can leave empty values in the array, use pop, push, shift, or unshift instead
concat() - creates a new array by concatenating existing arrays. Does not change existing arrays. Ex. arrayName1.concat(arrayName2)
splice() - splice can be used to add or remove elements from an array at specific locations. The first parameter defines the position where new elements should be added (spliced in). The second parameter defines how many elements should be removed. The rest of the parameters define the new elements to be added. Returns an array with the deleted items.You can remove items by not putting parameters that add new elements.
slice() - takes in two arguments that selects elements from the start argument, and up to (but not including) the end argument. Remember, array indexes start with 0. If the end argument isn’t there, the rest of the array is sliced.
Loops

For loop - for(let i=0; i < arrayName.length; i++){}. For loops take in a variable, a condition, and then what happens after one iteration of the loop. In this example i can be the index value of an array. The variable “i” is most commonly used in loops
For in loop - iterates until every value in an array has gone. Ex. for (variable in array){}
While loop - executes code while a condition is true. Don’t forget to have something in the loop that will make the loop eventually end..
Do while loop - do {} while (condition). You use this loop when you want the loop to be executed at least once regardless of whether a condition is true or not. Again, don’t forget to have something in the loop that will make the loop eventually end.
Chrome Dev Tools - right click and choose inspect to open, used to make changes without changing actual document
Screen size - click the button that looks like 2 devices to view your webpage at different sizes. You can also select a specific device from the dropdown menu.
Selector tool - chose the arrow with a box to select a certain element on the webpage and view it within your HTML document
Style panel - you can view, change, add, or disable styles of a selected element in the styles panel. It also has a visual representation of the box model for each element on the page
Console panel - shows what’s being logged to the console and will display errors within your Javascript code. Sometimes, console.log is used to see the values at different points in time during the program as a way of debugging.
Elements panel - You can edit the HTMl by right clicking and selecting “edit as HTML’
Plenty More - Chrome dev tools go way more in depth, these are just the basics. Here’s the link to the dev tool documentation: https://developer.chrome.com/docs/devtools/

Activities if we have extra time

1. Create an array and a function that takes all of the numbers in that array and adds them together.
2. Build upon greeting systems from last time by using an array of first names, last names, and birth years. Greet a total of 5 people. Answer:

```
let firstNames = ["Scott", "Barry", "John", "Susan",
"Bob"];
let lastNames = ["Bonds", "Jenkins", "Lewis", "Smith",
"Job"]
let currentYear = 2023;
let birthYears = [1918, 1945, 1976, 1982, 1972]
let fullNames = [];
let ages = []

for (let i = 0; i<firstNames.length; i++) {
  let fullName = firstNames[i] + " " + lastNames[i]
  fullNames.push(fullName)
```

```
}
for (let i = 0; i<birthYears.length; i++) {
  let age = currentYear - birthYears[i]
  ages.push(age)
}
for (let i = 0; i<fullNames.length; i++){
  let message = `Hello, my name is ${fullNames[i]} and I am
${ages[i]} years old`;
  console.log(message)
}
```

If you're looking for how to make JavaScript actually impact a webpage, google "manipulating the DOM."Make sure to add the defer keyword after where you link your Javascript to HTML.