**JavaScript in Action Worksheet**

* Inside your projects folder, create a new folder called ‘JavaScriptInAction’
* Open Visual Studio Code, and select ‘File > Open Folder’ at the top to open the JavaScriptInAction folder. In the Explorer, click the ‘new file button’ and create a file called “worksheet.js”. Applying the correct extension to the file name (.js) will create this file asa JavaScript file.

**Variables**

A Variable is the appropriate tool any time we want our program to remember a value that can be used later. Once a variable has been declared, it can be given a new value later to reflect the changing state of the application.

1. **Days of the Week**
   1. Create a variable called **dayOfWeek** and assign it a value of the string ‘Monday’.
   2. Log this variable to the console
   3. On the following line, change the value of **dayOfWeek** so that it has the value ‘Friday’.
   4. Log the statement ‘I can’t wait for Friday!” ***using the* dayOfWeek** ***variable***
   5. Put down a breakpoint on the first line of your code in the browser and run your code with the debugger. Use the Step Into button to go from line to line. Hover your mouse over the **dayOfWeek** variable every place it is used to see its current value change from Monday to Friday before it is printed.
2. **User Input**

The JavaScript **prompt()** function is similar to **input()** in that its purpose is to capture user input from the browser: It will display a message popup with an input field inside the browser window. When called, the **prompt()** function will **return** whatever is typed in the terminal after the message is displayed, and save it to a variable.

* 1. Create a variable called **animalInput** and set it equal to the prompt function call with the phrase ‘What is your favorite animal?’
  2. Create another variable called **colorInput** and set it equal to the prompt function call with the phrase ‘What is your favorite color?’
  3. Combine these two variables within a string so you can print the phrase “I’ve never seen a **<color> <animal>**!”

One of the most important things when creating a new variable is to think of a descriptive name that explains exactly what this variable represents in your code. If someone with no knowledge of your code were to jump in, they should be able to figure out what your variable represents.

Try not to abbreviate when creating variable names. The few keystrokes you save will not be worth the potential for confusion!

**Conditionals**

Before starting the next section, think of your favorite breakfast, your favorite lunch, and your favorite dinner. Write these three things in some comments in your code. Remember to start a line with **//** to designate it as a comment (hotkey shortcut: **CTRL** + **/** in Windows, **CMD** + **/** in Mac).

1. **Meals**
   1. Create a variable called **timeOfDay** and give it the value 1300
   2. Below that variable, create another and this time you choose the name. This variable will end up holding the value of one of the meals you have previously chosen, but right now, you don’t know which it will hold!
      1. (What would be a good name to use for a variable that might end up with any of those three values?)
   3. Now create a conditional with three parts:
      1. if **timeOfDay** has a value of less than 1200, assign your own variable the value of the breakfast you chose.
      2. If **timeOfDay** is between 1200 and 1700, assign your variable the value of the lunch you chose.
      3. If **timeOfDay** is greater than 1700, assign your variable the value of the dinner you chose
   4. After these conditionals, print your variable to the console. What value do you expect it to show?
   5. Go back to the line where you created **timeOfDay** and change its value twice so that you are able to see the other two meals print when you run the code.
   6. Now try to ‘break’ this section of code by giving **timeOfDay** a value that will cause an error. Think about how the variable is being used and what types of values would not make sense to be used that way. Run the code again to see the error get thrown.
   7. After you have successfully caused this section of code to throw an error, place a breakpoint on the first line of code in the file and debug the program. Watch how it will run all previous code until it encounters the line where the error occurs.
2. **Random Number**
3. Declare a variable to store a random number between 0 and 10. You will need to do some research to determine how to generate a random number in JavaScript.
   1. A good search term to use: “random number JavaScript”
4. If the number is between 0 and 2, print to the console “**Beatles**”
5. If the number is between 3 and 5, print to the console “**Stones**”
6. If the number is between 6 and 8, print to the console “**Floyd**”
7. If the number is equal to 9 or 10, print to the console “**Hendrix**”

**Loops**

In JavaScript, loops are an important tool that allow developers to execute a block of code over and over as long as a condition holds true. This worksheet will allow you to practice a for loop and while loop.

**For Loop**

A for loop is ideal in a situation where you know exactly how many times you want the loop to run.

A for loop consists of three parts:

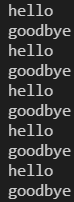
* Start – initial value for the iterator
  + i.e., **let i = 0;**
* Condition – the condition that exists to determine when the loop should continue iterating (as long as condition is true) and when the loop should complete (when the condition becomes false)
  + i.e., **i < 5;**
* Iteration – determines how much the iterator increases or decreases each iteration
  + i.e., **i++**

Graphical user interface, text, application

Description automatically generated

The above line of code creates a scope where any included code would be repeated 5 times. Additionally, the variable **i** that is created will increment from 0 to 4 with each loop.

1. Write your own for loop that will display the message “JavaScript is cool!” 7 times.
2. Write your own for loop that will display the numbers 0-10, one number per loop. You will need to utilize a variable for this!
3. Write a for loop that will print the following sequence:



**While Loops**

A while loop is ideal in a situation where you do not know how many times you want the loop to run. Instead, the loop will continue to iterate as long as the condition remains to be true. Once the condition becomes false then the loop completes.

Text

Description automatically generated

HINT: if **isInstructorAwesome** is set equal to true, make sure to have a way to set **isInstructorAwesome** equal to false somewhere inside the while loop to prevent an infinite loop from occurring

**If you get caught in an infinite loop, you can close the browser window to stop the process.**

**Functions**

1. **Favorite Movie (void functions)**
   1. Create an appropriately named variable and assign it the value of the name of your favorite movie.
   2. On the next line, print this variable to the console. Run your code to test it out.
   3. Now, move the previous two lines of code you wrote into a function named ‘**printMovieName**’
   4. Run your code again (be sure to invoke your function!!)
2. **Favorite Band (return functions)**
   1. Write a function that asks a user to enter their favorite band
   2. On the next line, the function should return the user’s response
   3. When you call the function, save the returned result as a new variable.
   4. After the function is called, print the returned result.
3. **Concert (parameters)**
   1. Create a new function called **concertDisplay.** This function should have one parameter called **musicalAct**
   2. Inside the function, ask the user to enter the street they live on and save it as a variable called **myStreet**.
   3. Print a string that combines the variable that holds the user’s response along with the **musicalAct** parameter. The message should end up saying “It would be great if <**musicalAct**> played a show on <**myStreet**>!”
   4. Now when you call the **concertDisplay** function, pass in the variable that was returned from the Favorite Band section.

**Arrays**

1. **Desktop Items**
   1. Create a new variable called **desktopItems**. Establish this variable as an array by setting it equal to a pair of square brackets [].
   2. When creating an array literal like this, we can insert some initial values at the same time it is created:
      1. Between the brackets, add a few strings to describe three items you see around you (‘desk’, ‘lamp’, etc.) Make sure to separate the items with commas **OUTSIDE** the quote marks.
   3. What index values do these 3 items have?
   4. Print to the console the item that resides at index 1.
   5. On a new line, use the JavaScript array method that allows you to add new things to an existing array to add the string ‘Infinity Gauntlet’ to your **desktopItems** array.
   6. Now, utilize a for loop to iterate over your array and print out each item one at a time.

**Magic Number**

This exercise will bring together all the pieces you have learned about so far!

1. Create a variable called **magicNumber** and give it an initial value of 50
2. Create a variable called **guess** and give it an initial value of 0
3. Create a while loop that will allow a user to continue to input a guess until they correctly choose the magic number.
4. If the number guessed is lower than the magic number, it should print “Too low!”
5. If the number guessed is higher than the magic number, it should print “Too high!”
6. If the number is within 10 of the magic number in either direction, it should print the appropriate message and additionally print “Getting warmer!”
7. If the number is correctly guessed, instead of “Getting warmer!”, print the magic number along with a congratulations message and break the loop.
8. Once you have tested and determined it works, go back to the beginning and change **magicNumber** to be assigned a random number between 0 and 100 instead of the hardcoded value.
9. Play your new guessing game a few times! Try to beat your own score
10. After playing a few times, put down a breakpoint at the start of the while loop and run it with the debugger. Hover over the **magicNumber** variable to see how much easier it is to determine the value of your variables by debugging this way!