Name _____ Period _____ Role (Circle one) Programmer/Driver

Name _____ Period _____ Role (Circle one) Programmer/Driver

Getting Started with Javascript

Your Tasks (Mark these off as you go)

- ☐ Create your first project
- ☐ Have Ms. Pluska check your first project
- ☐ Add comments
- ☐ Create and initialize variables
- ☐ Have Ms. Pluska check your comments and variables
- ☐ Apply arithmetic operations
- ☐ Apply concatenation to join variables
- ☐ Prompt the user for input
- ☐ Have Ms. Pluska check arithmetic operations, concatenation, and user prompts
- ☐ Complete challenges 1 thru 3
- ☐ Receive credit for the group portion of this lab
- ☐ Receive credit for the individual portion of this lab

☐ Create your first project

Create the required files

All modern browsers are capable to rendering javascript, which makes developing in javascript quick and easy.

- ➔ First create a new folder on your computer called IntroToJavaScript.
- ➔ Add two new files to this folder,
  - Index.html
  - App.js

In your Index.html file, add the following code

Index.html

```
<html>
      <head>
           <script src = "App.js"></script>
     </head>
</html>
```

Using the console

The console is a panel that displays important messages, like errors, for developers. Much of the work the computer does with our code is invisible to us by default. If we want to see things appear on our screen, we can print, or log, to our console directly.

In JavaScript, the console keyword refers to an object, a collection of data and actions, that we can use in our code. Keywords are words that are built into the JavaScript language, so the computer will recognize them and treats them specially.

One action, or method, that is built into the console object is the .log() method. When we write console.log() what we put inside the parentheses will get printed, or logged, to the console.

It's going to be very useful for us to print values to the console, so we can see the work that we're doing.

```
console.log(5);
```

The example above logs 5 to the console. The semicolon denotes the end of the line, or statement. Although in JavaScript your code will usually run as intended without a semicolon, we recommend learning the habit of ending each statement with a semi-colon so you never leave one out in the few instances when they are required.

To view the log simply right click on the webpage where your javascript is running, then select inspect. The console log will appear as a window in the developer tools menu.

- ➔ In your app.js page, write a line of code that logs your age. Refresh your index.html page in your browser. Then right click and select inspect. Locate the console window to see the output.

- ➔ On the next line, log a number representing the number of weeks you've been programming.

☐ Have Ms. Pluska check your first project

Before you continue have Ms. Pluska check your first project

Do not continue until you have Ms. Pluska's (or her designated TA's) signature _____

☐ Add comments

As we write JavaScript, we can write comments in our code that the computer will ignore as our program runs. These comments exist just for human readers.

Comments can explain what the code is doing, leave instructions for developers using the code, or add any other useful annotations.

There are two types of code comments in JavaScript:

- A single line comment will comment out a single line and is denoted with two forward slashes // preceding it.

  ```
  // Prints 5 to the console
  console.log(5);
  ```

  You can also use a single line comment to comment after a line of code:

  ```
  console.log(5);  // Prints 5
  ```

- A multi-line comment will comment out multiple lines and is denoted with /* to begin the comment, and */ to end the comment.

  ```
  /*
  This is all commented
  console.log(10);
  None of this is going to run!
  console.log(99);
  */
  ```

You can also use this syntax to comment something out in the middle of a line of code:

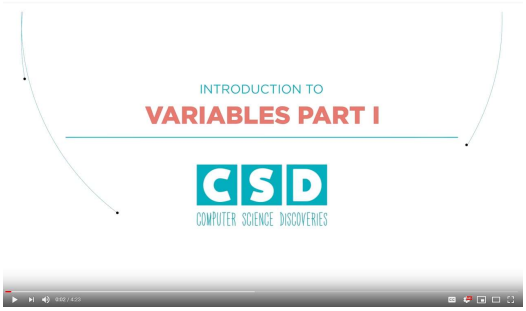
```
console.log(/*IGNORED!*/ 5);  // Still just prints 5
```

- ➔ On the first line of your app.js file, write a single line comment that says Opening line.
- ➔ Below this line use block quotes to write the following

```
firstname lastname
CS Special Topics
Introduction to JavaScript
Today's date
```

□ Create and initialize variables

Variables are data types that we use in programming. Variables are essential for controlling the memory in our programs. Watch the video below to learn more about variables,



- ➔ Copy and paste the code below into your App.js file,

```
var score;
score = 0;
console.log("The value of score is:");
console.log(score);
```

You're going to add another variable, assign it a value, and then display it to the console. Use the starting code as a model for what you need to create now.

- ➔ Create a new variable called lives
- ➔ Set the value of lives to be 3.
- ➔ Add a console.log messages to show the value of lives

It's so common to want to create a variable and give it an initial value, that JavaScript has a shortcut that lets you create and assign with one line of code like this:

```
var score = 0;
```

- ➔ Rewrite your code in your App.js file,
  - such that the variable score is declared and assigned with one line
  - such that the variable lives is declared and assigned with one line

The above examples illustrate how to store numeric data in memory. In javascript, we can also store String type variables. A String is any grouping of characters on your keyboard (letters, numbers, spaces, symbols, etc.) surrounded by single quotes: ' ... ' or double quotes " ... ". Though we prefer single quotes. Some people like to think of string as a fancy word for text.

- ➔ Copy and paste the code below into your App.js file,

```
var name;
name = "wigglesworth";
console.log("My name is ");
console.log(name);
```

- ➔ Change the variable name to your name
- ➔ Create a new variable called about, then assign a sentence about you to this variable. Do this on one line.
- ➔ Log the name and about variables to the console.

□ Have Ms. Pluska check your comments and variables

Before you continue have Ms. Pluska check your comments and variables

Do not continue until you have Ms. Pluska's (or her designated TA's) signature _____

□ Apply arithmetic operations

Basic arithmetic often comes in handy when programming.
An operator is a character that performs a task in our code. JavaScript has several built-in in arithmetic operators, that allow us to perform mathematical calculations on numbers. These include the following operators and their corresponding symbols:

- Add: +
- Subtract: -
- Multiply: *
- Divide: /
- Remainder: %

The first four work how you might guess:

```
console.log(3 + 4); // Prints 7
console.log(5 - 1); // Prints 4
console.log(4 * 2); // Prints 8
console.log(9 / 3); // Prints 3
```

Note that when we console.log() the computer will evaluate the expression inside the parentheses and print that result to the console. If we wanted to print the characters 3 + 4, we would wrap them in quotes and print them as a string.

The remainder operator, sometimes called modulo, returns the number that remains after the right-hand number divides into the left-hand number as many times as it evenly can: 11 % 3 equals 2 because 3 fits into 11 three times, leaving 2 as the remainder.

- ➔ Change a new variable called age and assign your age to this variable
- ➔ Inside of a console.log(), add 3.5 to your age. This is the age you'll be when we start sending people to live on Mars.

- ➔ On a new line write another console.log(). Inside the parentheses, take the current year and subtract 1969. The answer is how many years it's been since the 1969 moon landing.
- ➔ On a new line write another console.log(). Inside the parentheses, multiply 0.2708 by 100. That's the percent of the sun that is made up of helium. Assuming we could stand on the sun, we'd all sound like chipmunks!
- ➔ Create on last console log, print the number that remains when your age is divided by 10.

□ Apply concatenation to join variables

Operators aren't just for numbers! When a + operator is used on two strings, it appends the right string to the left string:

```
console.log('hi' + 'ya'); // Prints 'hiya'
console.log('wo' + 'ah'); // Prints 'woah'
console.log('I love to ' + 'code.')
// Prints 'I love to code.'
```

This process of appending one string to another is called concatenation. Notice in the third example we had to make sure to include a space at the end of the first string. The computer will join the strings exactly, so we needed to make sure to include the space we wanted between the two strings.

```
console.log('front ' + 'space');
// Prints 'front space'
console.log('back' + ' space');
// Prints 'back space'
console.log('no' + 'space');
// Prints 'nospace'
console.log('middle' + ' ' + 'space');
// Prints 'middle space'
```

Just like with regular math, we can combine, or chain, our operations to get a final result:

```
console.log('One' + ', ' + 'two' + ', ' + 'three!');
// Prints 'One, two, three!'
```

- ➔ Inside a console.log() statement, concatenate the two strings 'Hello' and 'World'
- ➔ Oops, we forgot about the space last time! This time, console.log() 'Hello' and 'World' again but this time make sure to use string concatenation to also include a space (' ') between the two words.

□ Prompt the user for input

Programs become even more interesting when we can interact with the user. A short way to ask a user for information is with the prompt command, which pops up a dialog box asking the user for input. Consider the example below. The example below prompts the user for their first and last name, then prints a message to the console.

```
var firstName = prompt('Enter your first name');
var lastName = prompt('Enter your last name');
var fullName = firstName + ' ' + lastName;
console.log('Hello' + fullName);
```

- ➔ Add the code above to your App.js file and run it
- ➔ Modify the code to prompt the user for more information. For example, “What is your favorite food?”, “What is the most interesting place you have visited?”.
- ➔ Print the user’s name, along with the additional information to the console. Make sure it is properly spaced and legible.

□ Have Ms. Pluska check off arithmetic operations, concatenation, and user prompts

Before you continue have Ms. Pluska check off arithmetic operations, concatenation, and user prompts

Do not continue until you have Ms. Pluska's (or her designated TA's) signature _____

□ Complete Challenges 1 thru 3

Challenge 1

In your App.js file, debug the following code segments

```
var apples = 7;
console.log(Aples);
var Oranges = 13;
console.log(oranges);
```

```
var score = 7;
console.log("The value of score is:");
console.log("score");
var lives = 20;
console.log (The value of lives is:);
console.log ("lives");
```

```
var firstName = "Wigglesworth";
console.log ("firstName");
var firstName = "Jug Head";
var lastName = "Jones";
console.log (firstName + lastName);
```

The code below, does not print what is expected. In comment block in your App.js file explain why

```
var firstNumber = prompt("Enter a number");
var secondNumber = prompt("Enter a another number");
console.log("The sum of the numbers are " + firstNumber + secondNumber);
```

Challenge 2

Create three variables – "problem1", "problem2", "problem3".

Assign the calculated value for each of the following to each problem respectively,

Problem 1: 79 + 3 * (4 + 82 – 68) -7 + 19
Problem 2: (179 + 21 + 10)/7 + 181
Problem 3: 10389 * 56 * 11 + 2246

Print the equation to the console, then catenate the answer to the end.

Challenge 3
Create a MadLibs game. Click on the link below to see an example,

https://drive.google.com/file/d/0B6lKrC3UayiJTEtZekNCOERXaU0/view?usp=sharing

Your game must Prompt the user for at least 10 different pieces of information

Your game MUST meet the following criteria,

- Collect all the information from the user first.
- Display the information in a story form, all at once to the console.

- The story must be displayed using a single console.log statement.

☐ Receive Credit for the group portion of this lab

Make sure to indicate the names of all group members, then submit this lab to the needs to be graded folder to receive credit for the group portion of this lab.

☐ Receive Credit for the individual portion of this lab

Implement challenges 1 thru 3 on your computer. Show Ms. Pluska the completed challenges to receive credit for the individual portion of this lab.