1.)What is Programming? (video notes:)

Programming is writing instructions for a computer to follow.

This typically involves moving, manipulating, and displaying data in a manner

that is useful to us as humans. To do this effectively, we need to realize that

as humans, we take advantage of how powerful our minds are. and don't often think of the tiny steps

into making any given decision. when we program, we have to think about what all the small steps are

to tell the computer exactly what to do one tiny step at a time.

for example, how many words are in the sentence, The cat jumped off the shelve and landed on my hand.

There are eleven words in this sentence, now how do we know that? we might say something like, because

i counted the words? but how do we know how to count words, what is actually is going on, is when we are counting the words

our brains are simplifying the process for us so we don't have to think about everything that goes into simply counting the words

in the sentence. but the problem is computers are not that smart. which means to write instructions for a computer to do something

we need to break down tasks into the simplest steps possible. within reason of course. take this word count for example.

we start by counting from 0, we look at each character, and when we see a space we increment our count by one. This is how we know

there is a new word to add to our total count because we see the space character. Think hard about the algorithmic steps of how we as humans

do what may seem simple tasks. This will help us a lot when we are learning how to program.

Programming is writing instructions for a computer that tell it how to move, manipulate, and display data in an automated fashion.

Steps to Programming:

Decide how to solve a task or request -- remember to break the task into smaller pieces to be solved

Write the code to solve the task by first writing code to solve each smaller piece of the solution

Compile and/or run the program that has been written.

Test and Debug the program

Test: Make sure the code words as designed

Debug: Track down and fix any flaws in the code that you wrote.

Document exactly what has been coded and how it works.

2.)Command Line Interface (CLI) (video notes:)

C:\Users\espin> Would be how the command line displays itself in a terminal

>dir displays all the directories in the folder

>ls displays all the directories in the folder

>dir [dir name] displayed directory of dir selected without having to be inside of directory

>cd [name of directory] change directories. (using tab will auto-complete path)

>cd.. To current directory parent directory

>cd../.. To go back 2 directories

>[full path] To directory of path entered

>cls clears terminal screen

>mkdir create directory [if includes spaces requires double ""]

>copy [item being copied] [destination being copied to]

copies file to destination directory

>copy test.txt testcopy.txt

copies file test.txt and names it testcopy.txt in the current directory

>del deletes file or folder

>move moves file to new directory

>echo "text to put into the file" > filename

creates a file with text inside of it.

3.)Variables, Data Types, and Operations (video notes:)

In programming we use things called variables to help us keep track of data. If you have apiece of data

lets say 56.95, how can we tell what that number represents? we can't. unless we have a label or name assigned to it.

So if we say our trash bill is 56.95 then that number would make more sense. We could then even just say trash bill and it would represent

the number without heaving the state the number again itself. This is how variables work. We assign a name to some data then anytime we

refer to that name, we are referring to the data it's assigned to which makes it much easier to keep track of and process data. Data

isn't just numbers though, it can come in different types, some others types of data include textual data like words and characters, Boolean

data such as true or false, and a few others. JavaScript is a dynamically typed language. That means that when we declare a variable we don't

have to explicitly say what type of data that variable will be representing. instead, JavaScript infers the data type based on the data. for example

it knows that first name would be a string if i assigned the value sam to it. and if I assign the value 10.50 to it, then it would know it's a number

and it would treat first name as a number. lets look at some of the most common data types in JavaSCript and how to declare a variable using them. to start a variable deceleration has a few parts, first we need to say that we are declaring a variable. to do that we use the var keyword. Next we give our variable a name. A variable name should be descriptive of what the data is that it's identifying. for example. if we want to have a variable that points to the balance in our bank account. a good variable name could be bankAccountBalance. This style of writing variable names where the first letter of the first word is lowercase and the first letter of each additional word is capitalized is called camel case and is the standard convention for variables in JavaScript. Next we assign a value to our variable using the assignment operator or equal sign. and then put the value on the right side of the operator. finally we end with a semicolon. semicolon are like periods in JavaScript, whenever you end a statement you want to put a semicolon. String data is denoted by single quotes or double quotes. Here are a couple of examples. var firstName = "Tom"; JavaScript knows that this is a string because it is in between double quotes. var lastName = 'Sawyer'; It also knows that this is a string because it is between single quotes. There is no difference between single quotes or double quotes other than the preference of the developers. numeric data is denoted without any quotes and using only numbers. the numbers can have decimals or be whole numbers. Here are a few examples. var numberOfFriends = 5; Here notice that there are no quotes around it. just a number and javaScript knows that because numberofFriends is assigned 5 any time we refer to numberOfFriends we'll be referring to that value 5. We already have another example here bankAccountBalance where we have a decimal. Boolean data is true or false. written complete out with no quotes. here are a couple or examples. var isHotOutside = false; var isHappy = true;

again, notice there are not quotes around it like our strings, false and true are both keywords in JavaScript used for Boolean values. We will learn more about data types in later courses. but for now knowing how to declare a variable and knowing that JavaScript infers the data types based on the assigned data is the most important thing to understand.

Variables:

Since programming is moving, manipulating, and displaying data, we need a way to know what data we are working with. We need a way to assign names to data. To do this, we use something called variables.

Imagine that we have a piece of data: 25.74 How do we know what that data is? What it represents? It could represent the price for an item, the balance of a bank account, the distance from one place to another. It could represent any number of things. Variables allow us to identify what specific data is so that we can refer to the data by its variable name and write instructions telling the computer what to do with it.

Data Types:

Variables can refer to different types of data. Different types of data are typically used in different ways. For example, alphanumeric/textual data is typically used to label something (think anything with text), while numeric data deals more with values and math.

Below are some data types that JavaScript uses:

Boolean True or False

Number Numeric values

String alphanumeric/text values

Variable and data types:

In JavaScript we don’t have to tell a variable what type of data it will hold, the data type is determined implicitly based on the data assigned to the variable. This is called dynamic typing. JavaScript is also loosely typed, meaning that the data type of a variable can change (i.e. a variable could be pointing to a string and then changed to hold a number value).

To declare variables in JavaScript, we start with the keyword var. This lets JavaScript know we are creating a new variable. We then give the variable a name, or identifier. This is how we can refer to the variable after it’s been declared.

String Example:

For example, if we have customer data we are working with, we may have a variable called customerFirstName that holds String data representing the first name of a customer. After declaring the variable name/identifier, we use an assignment operator (the equal sign: =) followed by the value we want to assign to the variable. Finally, we end the statement (the line of code) with a semicolon. The result is as follows:

Var customerFirstName = “Sam”;

That is how we declare a variable. Anywhere we refer to customerFirstName name after that line, the computer will substitute the value “Sam”.

Number Example:

Below are some examples of variables that have Number values assigned to them.

Var bankAccountBalance = 100.54;

Var numberofFriends = 10;

Notice how there are no quotes around the number values. Strings are denoted with single or double quotes, but numbers do not use quotes.

Boolean Examples:

Below are some examples of Boolean values assigned to variables.

Var isHotOutside = false;

Var isHappy = true;

Notice that the Boolean values true and false also do not have quotes around them.

Camel Case:

When we write variables, notice how the first letter of the first word is lowercase and then each first letter of each following word is uppercase. This is a naming convention called Camel Case, or camelCase to follow its own convention, and is the naming style we use for variables in JavaScript.

Connecting a JavaScript File to an HTML file:

In order to test much of the code we are going to write, we need to run it in a web browser. To do this, we create folder that will house our JavaScript files and an HTML file that we will open in the browser. We will link our JavaScript file to our HTML file so that when the HTML file loads it runs the JavaScript.

To do this, we need to add a script tag to the head element in our HTML.

Below is an example, but the index,js would be replaced with whatever the name of our JavaScript file is.

Note: JavaScript files need to end in a .js extension and HTML files end in a .html extension, meaning that a proper JavaScript file name would look like: myFile.js and a proper HTML file name would look like: myFile.HTML.

Example: Index.html

<html>

<head>

<script src=”index.js”></script>

</head>

</html>

When we open this HTML file in the browser, the JavaScript file mentioned in the scipt tag will run.

4.)Operations (video notes:)

Giving a name to data is not enough to make a useful program. We need to perform operations on data. Operations consist of an operator that acts on operands. Most commonly two operands. Before we start practicing some operations lets set up our files so that we can see the results of our operations. We will be running our JavaScript in the browser. So we will create an HTML page that will load in the browser and link our JavaScript file to it. Lets start by creating the html file. I’m going to call it index.html. Don’t worry about all the details of the HTML we are about to write. That’s beyond the scope of this lesson but we will be using it to run our JavaScript in the browser. The only important thing to know here is that you are adding your script and the source is going to point to whatever the file name is for where you’re writing your JavaScript. You will need to install the browser add on. Named open in browser. So I have mine installed right here. Make sure you have this installed. And then we can go ahead and right click and say open in default browser. And that will launch our browser. Right now nothing is happening because we don’t have anything coded any operations actually happening here on our JavaScript file, but we are about to add those operations. So lets start just to make sure that everything is working, we are going to run console.log we’ll go ahead an put a string in there and do hello world. Console.log(“hello world”); So now if I go to my HTML file, and right click and open in default browser. Make sure if you don’t see open in default browser after you installed it make sure you are not clicking on the JavaScript file You have to click in the index.html file on the HTML file. Open that on our default browser. This will pop back up. And then we can open our dev tools, by either clicking f12 or by right clicking on the browser and selecting the inspect button, and then we can look at the code that was printed to the console. Right here we’re on elements go over to the console, there we go, we printed hello world. So that means that our JavaScript file is accurately connected to our HTML file. So now we can write some operations and test the results of those right here in our console. So go back to vs code. The most common operators that you are already used to seeing are mathematic or arithmetic operators such as addition, subtraction, multiplication, division. Lets take a look at how these operators work with numeric data and variables that point to numeric data. So we are going to use console.log for all of these. Console.log and our operations don’t have to go inside of a console.log or any other function or method call. We are just using this to see the results of the operations. So we’ll start with 5 + 3. And as I mentioned earlier. An operations is when an operator acts on operands. So this an operator. The addition operator. To the left and right of this, these numbers are the operands and then the operation will be the result here. 5 + 3. console.log(5 + 3); So we’ll do that one we’ll do a couple more before we switch over to our browser and look at them. So that’s addition, lets do 4 – 2 for subtraction. Console.log(4 - 2);. We’ll do 6 \* 7 for multiplication. Console.log( 6 \* 7 ); and lets do 8 / 2 for division. Console.log( 8 / 2). So if we come back to our browser notice nothing shows up. You have to refresh the browser to reload that page and re run the script. So you refresh it then we can see we have 5 + 3 = 8 I think 4- 2 = 2 and 6 \* 7 = 42. And 8 / 2 = 4. And we got the proper results that we would expect. We could also use variables. So lets do console.log(bankAccountBalance – 30); and then lets do another one console.log(bankAccountBalance \* numberofFriends); Save that always make sure you save. Refresh. And we can see the bankAccountBalance - 30 = 22.1 and then we can see bankAccountbalance \* 5 = 260.5 which was the numberOfFriends. Now notice that \* 5 it’s not 22.1 \* 5 it’s the value of bankAccountBalance, we never changed the value of BankAcccount, we just printer out whatever that value was – 30. Other types of operators in addition to mathematic operators include equality, logical and assignment operators. We are already familiar with the assignment operators it is the single equal sign = used to assign the value to a variable, and we will talk more about logical operators in another lessons. But let’s talk a little bit about equality. The equality operator consist of two equal signs == and is used to determine if two values are equal. The result of an equality operation is a Boolean value either true or false. Lets take a look at a couple of examples. Console.log(5 == 5); So that double equal there that is our equality operator. If we save this and refresh we get true. Because 5 is indeed equal to 5. Lets see if we do 5 == 6. Save it refersh. It’s false because 5 does not equal 6. And just like any other operation and the value variable we can use variables. So lets say our bankAcountBalance equal to our numberOfFriends and we know that the answer is no, That is false. Console.log(bankAccountBalance == numberOfFriends); We can also compare again the variable to a value, so console.log(bankAccountBalance == 52.10); Save that and refresh and we see the very last one is true.

Operations:

Knowing that there are different types of data is great, but what is data good for if we don’t use it in some way? For example, point of sales systems (the software used when we purchase something from a store, restaurant, etc) have to add up line items and then apply a tax to them. That means we have to perform actions on data (addition and multiplication in this case). In programming, these actions are called operations.

An operation consist of one or more pieces of data, known as operands, and an operator, and performs a calculation or action on the operands thus resulting in a new value. One operator we are already familiar with is the assignment operator (the equal sign =), which assigns the data on the right-hand side to the variable name/identifier on it’s left. The data on the right and the variable on the left are the operands in this operation.

Var name = “Sam”;

While operations take one or more operands, most operations take exactly two operands. For example, our arithmetic operations (additions, subtraction, multiplication, and division) take two number operands and perform their related math operation on them. To see how some of these work, we can use the console.log() function, which prints values to the console when executed. To test this out, we need to make sure we are writing out code in a JavaScript file that is linked to an HTML that we will open in our browser.

Console.log(5 + 3);

Console.log(4 – 2);

Console.log(6 \* 7);

Console.log(8 / 2);

The above operations provide examples of addition, subtraction, multiplication, and division in that order. The operation consists of the numbers and operator inside the console.log() function. The console.log() function is not part of the operation, rather a statement to print out the results of the operation to the console where we can see them.

Operations also work with variables, not just hard-coded values. For Example:

Var bankAccountBalance = 100.24;

bankAccountBalance = bankAccountBalance – 5;

console.log(bankAccountBalance);

In the above example, we initialize the bankACcountBalance variable with the value 100.24 using an assignment operator. On the next line, we use the assignment operator again to reassign the value of another operation bankAccountBalance – 5, which subtracts 5 from the original value and updates the variable. Note that there are two operations happening on that line.

Short-hand Notation:

As developers, we love making things short. Another way to write the second line in example above is this:

bankAccountBalance -= 5;

This is the same thing as saying BankAccountBalance = bankAccountBalance – 5 there is also the += operator for adding to a variable. Short-hand makes it easier!

Addition: numbers vs Strings:

The + operator can do different things depending on whether the operands are numbers or Strings. If the Operands are numbers, then the + operator performs addition. If any of the operands are Strings, it performs something called concatenation, which basically adds the strings together by combining them.

For example:

1.Addition:

Console.log(5 + 3); // logs out 8

2.Concatenation:

Console.log(“Hello” + “World”); // logs out HelloWorld

The + operator adds the values in the first example (addition for numbers) and concatenates them in the second example (concatenation for strings).