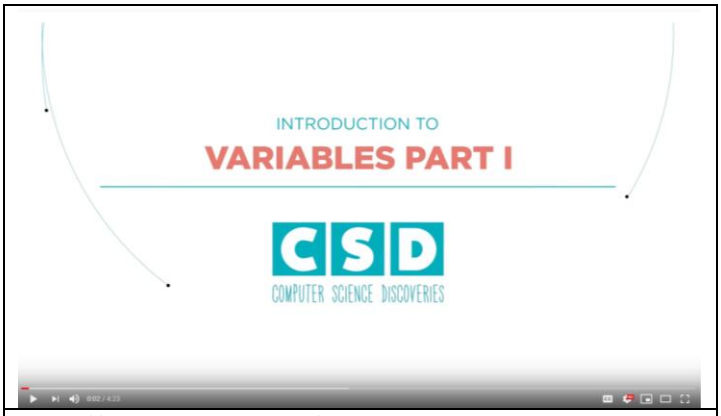
Set 17: Variables and User Input

Skill 17.01: Create and initialize variables

Skill 17.01 Concepts

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.

.



https://www.youtube.com/watch?v=G41G_PEWFjE

Variables in JavaScript are declared using the *var* keyword.

```
var x;
```

The variable we declared above is currently empty, that is no value has been assigned to it. Assigning a value to a variable is called initializing. The following code illustrates how to initialize the variable x.

```
x = 10;
```

Notice in the code above,

- We did not type var again. Once a variable is declared, it cannot be declared again. Leaving *var* in front of our variable x would actually generate an error.
- Also, notice that the variable name is always on the left of the equal sign. The value you want to assign to the variable goes on the right of the equal sign.

The value of a variable can also be changed at any point in the context where it was declared. For example, the code below, would change the value of x to 5.

```
x = 5;
```

Variables can also be manipulated. For example, we could add 1 to variable x as follows,

```
x = x + 1;
```

When naming variables, regardless of the type, the following rules apply,

- Variable names must begin with a letter (or an underscore character)
- Variable names cannot contain spaces
- The only "punctuation" character permissible inside a variable name is an underscore "_".
- Variable names cannot be one of the reserved words that are part of the JavaScript library

Below are some examples of legal and illegal variable names,

Legal	**Illegal**
Agro	139
D	139Abc
D31	Fast one
Hoppergee	Class
hopper_gee	slow.sally
largeArea	var
computerScience	Computer:Science
codeIsCool	code-is-cool

To avoid illegal variable names, you should always apply the "lower camel case" naming convention. That is, the first letter in the variable is lower case. If there are multiple words associated with the variable name, they are run together. Each word in the variable (except for the first word) is capitalized. Consider the example below.

```
var myFinalScore;
myFinalScore = 10;
```

To print the value of a variable to the console, we use the same print statement we learned previously. The following code could be used to print *myFinalScore*. Notice in the below example, there are no quotes around the variable. If there were quotes around myFinalScore, "myFinalScore" is what would be printed. Leaving the quotes off, prints the value of the variable.

```
console.log(myFinalScore);
```

It is 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 myFinalScore = 10;
```

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. Consider the following examples

```
var name;
name = “Wigglesworth”;
console.log("My name is ");
console.log(name);
```

Skill 17.01 Exercises 1 thru 4

Skill 17.02: Apply arithmetic operations

Skill 17.02 Concepts

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:

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
%	modulus

The first four work how you might guess:

```
var x = 3, y = 4, z = 1;
console.log(x + y); // Prints 7
console.log(y - 1); // Prints 3
console.log(y * x); // Prints 12
console.log(y / x); // Prints 1.25
```

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.

Skill 17.02 Exercise 1

Skill 17.03: Apply concatenation to join variables

Skill 17.03 Concepts

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'
```

Numbers and String expression can be combined, but be careful! Consider the following,

```
console.log(1 + 2 + 3 + ' One' + ', ' + 'two' + ', ' + 'three! ' + 1 + 2 + 3);
//prints 6 One,two,three! 123
```

Notice in the above example the numbers 1, 2, and 3 in front of the String expression were added together. However, the numbers 1, 2, and 3 after the String expression were not treated as numbers and were not added together.

Skill 17.03 Exercises 1 & 2

Skill 17.04: Cast a non-number to a number

Skill 17.04 Concepts

As you continue to evolve as a JavaScript developer, you will experience confusing moments. Dealing with NaN will undoubtedly be one of them.

NaN stands for Not a Number. In a previous example we encountered this error,

```
console.log(" Blah "+  10-5 + " Blah Blah " + 10-5);//printed NaN
```

The reason for this error in the example above is that JavaScript thinks you are trying to include a non-numerical phrase in a mathematical operation.

In fix the above example requires that we force JavaScript to treat 10-5 as a numerical operation. To this requires that we *cast* it to a number.

```
console.log(" Blah "+  Number(10-5) + " Blah Blah " + Number(10-5));// 5 Blah  Blah Blah 5
```

Skill 17.04 Exercise 1

Skill 17.05 Prompt for user input

Skill 17.05 Concepts

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);
```

Skill 17.05 Exercise 1