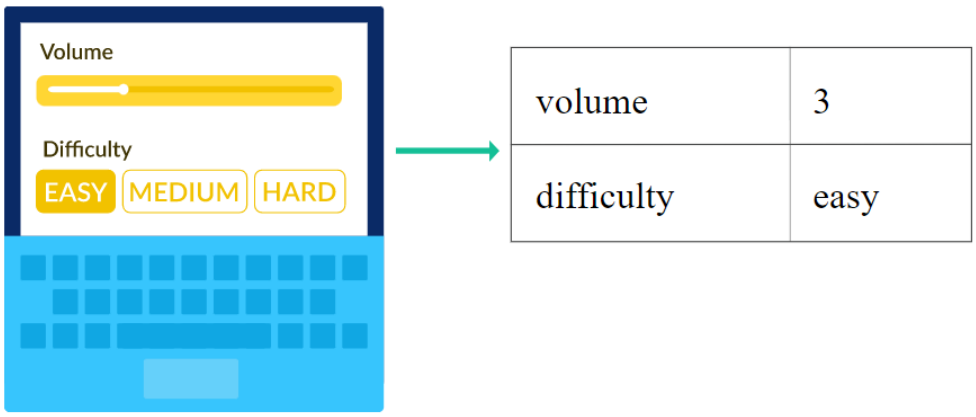
Set 19: Variables and User Input

Skill 19.01: Explain the purpose of variables

Skill 19.01 Concepts

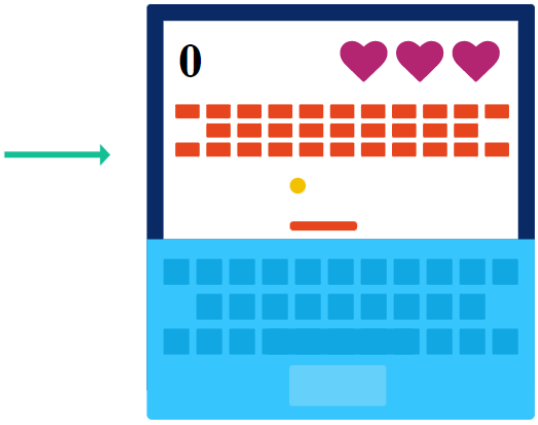
Computer programs instruct computers how to process data.

Sometimes the data comes from a user, like when a game asks you to pick a difficulty level and music volume,



Other times, the data is already stored in the program, like when that same game starts off with a score of 0 and 3 lives,

score	0
lives	3



Either way, the program can store that data in **variables**. Each variable has a name, a value, and a type. The value might change over time, and that's why its "variable."

That game in the previous example is using at least four variables,

name	value	type
`volume`	3	number
`difficulty`	`"easy"`	string
`score`	0	number
`lives`	3	number

Many variables store numbers and strings, like the ones above. Variables can also store other types of data, like lists, dictionaries, and Boolean values (true/false).

We'll start by learning numbers and strings, and dive into advanced types later.

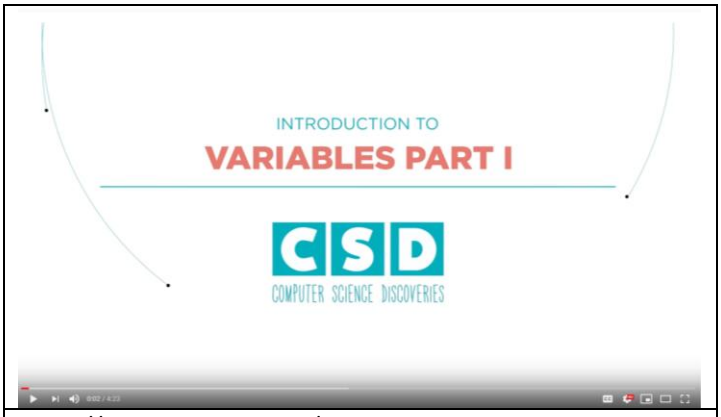
Skill 19.01 Exercise 1

Skill 19.02: Declare and initialize variables

Skill 19.02 Concepts

Watch the video below to learn about what it means to "declare" and "initialize" variables in JavaScript.

.

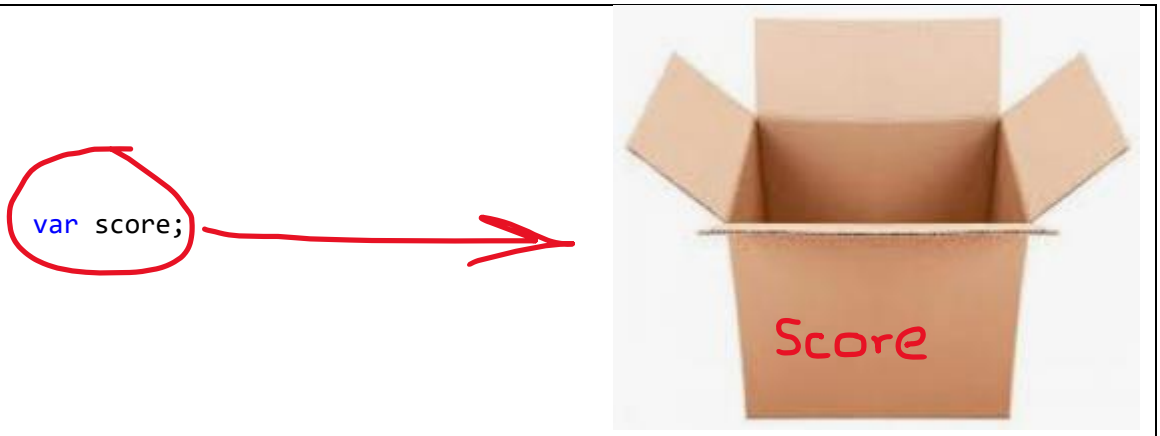


https://www.youtube.com/watch?v=G41G_PEWFjE

As mentioned in the video, declaring a variable in JavaScript is done with the *var* keyword. Below is an example of how to declare a variable named "score",

```
var score;
```

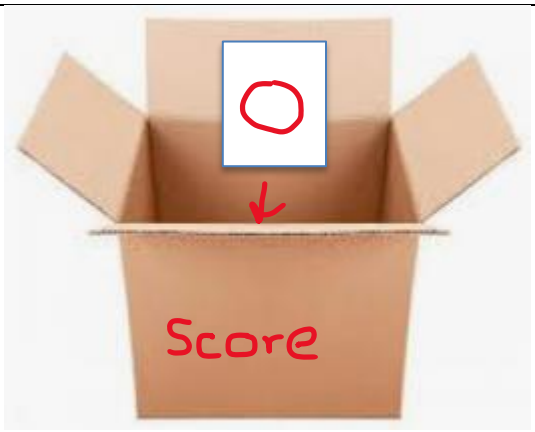
When you **declare** a variable in JavaScript you are creating a place holder in memory. You can think of this like an empty box.



The statement above creates an empty box named "score"

Notice our variable above doesn't contain a value. When we give score a value we are **initializing** it. Notice in the example below, we did not use the keyword var again - doing so would create a new box. To give score a value we use the equal sign followed by the value you want to assign to score.

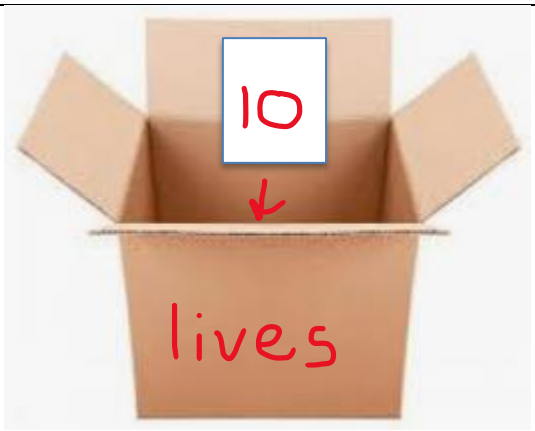
```
score = 0;
```



The statement above puts 0 in the score box.

It is also possible to declare and initialize a variable all at once,

```
var lives = 10;
```



The statement above creates a lives box and puts 10 inside of it.

Below are some key points about declaring and initializing variables,

- Variables are declared using the keyword var.
- The keyword var creates a place holder in memory for the variable. Every time you type var a new place holder is created.
- Variables are initialized by using an equal sign. The value you want to assign to the variable goes on the right of the equal sign.
- Variables can be declared and initialized with one statement

Skill 19.02 Exercise 1

Skill 19.03: Apply proper naming conventions to variables

Skill 19.03 Concepts

If you want to **give a name to a variable**, you must follow some strict rules:

- the name of the variable must be composed of upper-case or lower-case letters, digits, and the character _ (underscore) – the underscore is the only permissible character
- the name of the variable must begin with a letter or an underscore
- variables names cannot contain spaces
- upper- and lower-case letters are treated as different (a little differently than in the real world – *Alice* and *ALICE* are the same first names, but in JavaScript they are two different variable names, and consequently, two different variables)
- the name of the variable must not be any of JavaScripts's reserved words (the keywords – we'll explain more about this soon).

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

The JavaScript style guide recommends the lower camel case convention for naming variables. 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;
```

When you name variables, you should also give them concise meaningful names. Consider the examples below,

Variable name	Feedback
`var x = 10;`	Should be more descriptive
`var score = 100;`	Good.
`var y = "Aiden";`	Should be more descriptive
`var name = "Aiden";`	Good.
`var theNameOfThePersonWritingThisProgram = "Sofia";`	Way too long

The following keywords play a very special role in every JavaScript program

abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*
in	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

They are called **keywords** or (more precisely) **reserved keywords**. They are reserved because **you mustn't use them as names**: neither for your variables, nor functions, nor any other named entities you want to create. The meaning of the reserved word is **predefined**, and mustn't be changed in any way.

For example – **you can't name** your variable like this:

```
var import;
```

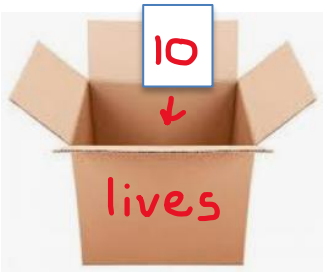
Skill 19.03 Exercise 1

Skill 19.04: Store integers in variables

Skill 19.04 Concepts

Unlike some other languages, JavaScript has no command for declaring a variable type. All variables regardless of the type are declared using the var keyword.

```
var lives = 10;
```



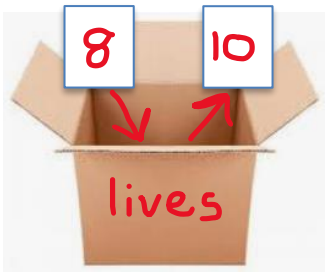
To check the value of a variable you can print it to the console,

Code	Output
`console.log(lives)`	`10`

You can change the value stored in a variable at any time by entering another assignment statement. For example,

```
var lives = 10;   //lives has a value of 10
lives = 8;        //now lives has a value of 8
```

The value of lives was originally 10, but it was replaced or overwritten with the value of 8. This process of overwriting is illustrated below,



Notice we did not re-declare lives with the var keyword when we overwrote it, doing so would create a new variable and would produce unexpected results.

Skill 19.04 Exercise 1

Skill 19.05: Apply arithmetic operations to variables

Skill 19.05 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
**	exponentiation
/	division
%	modulus

The first five 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 64
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 19.05 Exercise 1

Skill 19.06: Interpret number and string variable types

Skill 19.06 Concepts

Numbers

Integers (or ints for short) are whole numbers such as 4, 99, and 0. Floating point numbers (or floats for short) are fractions or numbers with decimal points like 3.5, 42.1, and 5.0. In JavaScript, integers and floats are considered numbers.

Just as we saw in previous examples, arithmetic operations can be applied to numbers and the result can be either stored in a variable or printed.

Code	Output
`var result = 8 * 3 / 2 + 2 + 7 - 9` `console.log(8 * 3 / 2 + 2 + 7 - 9)`	12

Strings

In JavaScript, text values are called strings. Just like integer or float values, you can store strings in variables. In code, string values start and end with a single quote (') or double quote(").

```
var spam = 'hello'
```

The quotes tell JavaScript where the string begins and ends. They are not part of the string value's text. If you print *spam* to the console you will see the contents of the spam variable. Remember, JavaScript evaluates variables to the value stored inside the variable. In this case, this is the string 'hello'.

Code	Output
`var spam = 'hello'` `console.log(spam)`	hello

Strings can have any keyboard character in them and can be as long as you want. Below are all examples of string variable types,

```
var a = 'hello'
var b = 'Hi there!'
var c = 'KITTENS'
var d = '7 apples, 14 oranges, 3 lemons'
var f = 'Anything not pertaining to elephants is irrelephant.'
var g = 'A long time ago, in a galaxy far, far away...'
var h = 'O*&#wY%*&OCfsdYO*&gfC%YO*&%3yc8r2'
```

Skill 19.06 Exercise 1

Skill 19.07: Apply concatenation to join variables

Skill 19.07 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 expressions 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 19.07 Exercise 1

Skill 19.08: Cast a non-number to a number

Skill 19.08 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 19.08 Exercises 1 and 2

Skill 19.09 Prompt for user input

Skill 19.09 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 19.09 Exercise 1

Skill 19.10: Declare and initialize variables in pseudocode

Skill 19.10 Concepts

The pseudocode below represents assigning a variable,

```
a ← expression
```

Whenever you see that pseudocode, it means that the variable a (or whatever it's called) is assigned the value of expression.

For example, you might see pseudocode like this:

```
age ← 21
```

That means the variable age is assigned the value 21.

Let's look at the equivalent code in a few textual languages.

language	code
JavaScript	`var age = 21;`
Python	`age = 21`
Java	`int age = 21;`

Skill 19.10 Exercise 1