

There are 4 ways to declare a variable, to store it in a small memory space. You can assign a value to the variable using '=', like this:



```
1 // Declares a variable named school
2 let school
3
4 // Assigns the value 'ESMAD' to the variable
5 school = "ESMAD"
```

To be concise, you can combine the declaration and assignment on a single line:



```
1 // Declares a variable named school and assigns a value to it
2 let school = "ESMAD"
3
4 // You can also declare multiples variables on a single line
5 let school = "ESMAD", city = "ESMAD", age = 34
```

The value type of a variable can change during the execution of a program and JavaScript takes care of that automatically



```
1 let price = 5
2
3 price = "car"
4
5 // Prints in the console 'car'
6 console.log(price)
```

This feature is called **dynamic typing**

A variable should be declared only once

A repeated declaration of the same variable is an error:



```
1 let message = "This"  
2  
3 // repeated 'let' leads to an error  
4 let message = "This"    // SyntaxError: 'message' has already been declared
```

Good example:



```
1 let message = "This"  
2  
3 message = "That"    // variable message will store the string 'That'
```

To declare a constant (immutable) variable, use `const` instead of `let`:



```
1  const age = 10  // the variable age will store the value 10
```

They cannot be changed. An attempt to do so would cause an error:



```
1  const age = 10
2  age = 20      // error, it is not possible to re-assign a constant value!
```

When a programmer is sure that a variable will never change, he must declare it with `const` to ensure and clearly communicate that fact to everyone

There are constants that:

- are known before execution (as a hexadecimal value for red)
- are calculated at run time, but are not changed after the initial assignment



```
1  const pageLoadTime =          // time pages takes to load
```

Var :

In JavaScript, the var keyword is used to declare variables. Variables declared with var have function scope, which means they are accessible anywhere within the function in which they are declared. Var is not the best option nowadays.

Naming :

There are two limitations to variable names in JavaScript:

- The name must contain only letters, digits or the symbols \$ and _
- The first character cannot be a digit

Examples of valid and invalid names:



```
1 let userName // good example
2
3 let 1a // cannot begin with a digit
4
5 let test123 // good example
6
7 let my-name // hyphens '-' are forbidden in variable names
```


There are some good habits in the chosen name to declare the variable to make the code more readable and easier to understand:

- Variable names are **case sensitive** (Upper and lower case)

The **banana** and **BaNAna** variable names are 2 different variables

- Reserved words

There is also a list of reserved words, which cannot be used as variable names because they are used by the language itself

For example: **let**, **class**, **return** and **function** are reserved

- Be descriptive, choose names that clearly describe the purpose of the variable. Avoid using generic names like 'a', 'b', 'x', etc.

- Use camelCase: where the first word starts with a lowercase letter and the following words with a capital letter.
Example: `(userName)` or use `snake_case` where the first word is separated from the second by a “_” (underscore) example `(user_name)`
- Avoid unnecessary abbreviations that can make the code more difficult to understand, use only if they are common and well understood in the context of the project.
- Be consistent, keep the same naming convention throughout the code, if you declared a variable in English, keep all variables with the name in English.
- While it's important to be descriptive, avoid overly long names.

Scope :

The scope of a variable is the context where it was defined


- Naming variables via `let` or `const` restricts the variable's access to the nearest surrounding block
- A JavaScript block is represented by `{ ... }`



```
1  let y = 2
2  {
3      let x = 3
4      console.log(x) // 3
5  }
6  console.log(y) // 2
7  console.log(x) // error: Uncaught ReferenceError: x is not defined
```

Data types:

- Booleans: used for true or false values;
- Strings: used for text values;
- Number: used for numerical values;
- Null: used to indicate the intentional absence of any object or value;
- Undefined: used to indicate that a variable has been declared but has not been assigned a value;



```
1 let fullName = true; // Boolean
2
3 let firstName = "Helen"; // String
4
5 let age = "31" // Number
6
7 "empty value" // Null
8
9 let arrayOfNames; // Undefined
10
```

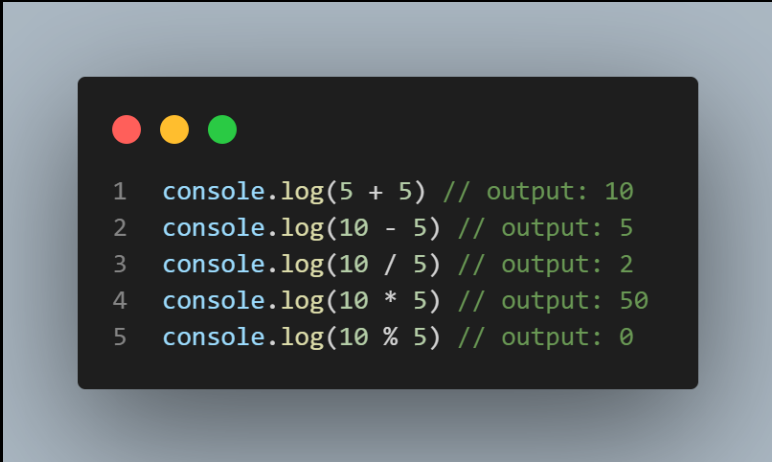
Operators:

The '+' is a special operator used to concatenate (join) strings or add depending on the declared variable type. That's why it's important to pay attention to the declaration of variables in JavaScript.

The remaining operators are used for arithmetic operations:

- "-" : Subtract
- "*" : Multiply
- "/" : Divide
- "%" : Remainder of division
- "++" : Increment
- "--" : Decrement

Finally, to print messages and values in the browser's console, `console.log()` is used



```
1 console.log(5 + 5) // output: 10
2 console.log(10 - 5) // output: 5
3 console.log(10 / 5) // output: 2
4 console.log(10 * 5) // output: 50
5 console.log(10 % 5) // output: 0
```