



JS | JavaScript Intro

Learning Goals

After this lesson you will be able to:

- Explain what JavaScript is and a bit of its history
- Explain what ES6 is and how JavaScript evolves
- Understand and apply basic JavaScript syntax
- Declare variables in JavaScript
- Properly name variables

What is JavaScript?

According to the official **MDN** (Mozilla Developer Network) documentation, **JavaScript can be defined as a *lightweight interpreted programming language with first-class functions*.**

Sounds super scary, we know, but let's break it down a bit:

- **lightweight programming language** is any programming language that is structured in such way to require very small memory usage while running. If you would like to learn more about this subject, you can start [here](#);
- **interpreted programming language** is any programming language that executes the program directly translating each line of code into [machine code](#). At this very moment this is enough to know but for deeper understanding, we suggest you start [here](#);
- a **programming language with first-class functions** according to the MDN means that in this language functions are treated like any other variable and this being said, they can be passed as an argument to other functions, can be returned by another function and can be assigned as a value to a variable.

To summarize this above, we can say:

JavaScript is a lightweight programming language that gets converted into machine language line by line and uses functions.

It is mostly used for Web pages (runs in browsers) but at the same time there are a lot of non-browser environments where it's used, such as Node.js (which you will learn how to use later on), among the others.

To simplify this:

JavaScript was created to run in every browser, but now you can run JavaScript in your server. This means you can **create a whole application using just the JavaScript language!**

😞 *Potential Interview questions* (each property clickable and will take you to external resources, feel inspired to explore):

JavaScript is a **prototype-based, multi-paradigm, dynamically typed language**, supporting **object-oriented, imperative and structured, and declarative** (e.g. functional programming) styles.

JavaScript was created in 1995 and as you might know, doesn't have anything with Java - the original name of JavaScript was *LiveScript*. Since 1995 up to this date, JavaScript went through a lot of updates and changes.

ECMAScript

Since the language evolved a lot, there had to be some kind of *standard* to make everything that was using JavaScript still running with all these updates.

This is where **ECMAScript** comes in the play.

ECMAScript is the standard and JavaScript is its most popular implementation.

The versions of ECMAScript (shorter *ES*) that are still in use are **ES5** (from 2009.), **ES6** (from 2015.), **ES7** (2016.).

In this curriculum, you will use mostly **ES6** and later JavaScript features.

This intro was very excessive but it's needed to start thinking about these terms from the very beginning. However, fully understanding of it will come as you progress through the course so don't stress, everything is under control ✅.

Okay, it was enough of theory, let's move to the real JavaScript and start writing some code.

Basic JavaScript Syntax

The **syntax** of a programming language is the **set of rules** that needs to be respected by programmers (who write the code) to be successfully interpreted by machines (that execute that code).

JavaScript syntax is loosely based on the Java syntax. This means quite a few **curly braces** { and **parentheses** () .

The semicolon is not mandatory in 99.5% of cases. Parentheses and curly braces ***are mandatory*** and will cause an error if they're left out. Be mindful of starting/ending these in the right place.

Output

We will use a function called `console.log()` to create an output. You will get used to writing a lot of `console.log()` and it will become your veeeerrryyyy good friend over time. Today we will practice in `codepen`, a site for writing simple JavaScript.

```
console.log("My name is Bob, and I'm writing JavaScript");
```



Click the **edit on codepen** link to play with the code on the website

Comments

Writing comments in your code is extremely important for multiple purposes:

- it will help you while learning (now), and to be able to quickly continue where you stopped after a longer break (later, at work)
- it will help you to easily skim through the code when trying to find something
- sometimes we just want to comment out a piece of code we don't need at the moment

To comment out something that fits in one line, we use `//`:

```
// a short, one-line comment
```

If the comment is on more than one line, then we need to use `/* */`:

```
/* this is a long, multi-line comment  
   Hopefully one day I'll appreciate  
   writing all these comments :) */
```

Now, let's start!

Variables

What is a variable?

The main purpose of variables is to store some information. We can think of them as of some kind of containers or storages that hold some data, which is going to or can be used at any or just at some points in our program. Very important thing is that these storages are **named**, or you can often hear **labeled**.

Variables can be seen as named (labeled) storages that hold some **values**. The purpose behind naming them is actually to be able to **reference** them later on.

For us will be very interesting to talk about *different kinds of values* that can be stored in variables, and these are known as **data types in JavaScript**: a string, a number, an array, an object, etc.

Declaring a variable in JavaScript

Before you can use a variable in a JavaScript, you have to **declare** it. The declaration of a variable doesn't mean that some value is assigned to it; it simply means "saving a spot in a memory" that will be later on filled with some value.

In order to **declare or define** a variable we can use keyword `let` or `const` and in older versions of JavaScript, you would find keyword **var** which is used to declare variables. `var` is still very much present and it's super important to understand *similarities and differences between `let` and `var`* and we will go over this a bit later. For now, just keep in mind that you could **in most cases use either `let` or `var` to declare the same variable**.

Variable declaration with `let`

For now, let's start declaring variables using the keyword `let`:

```
let name; // <== this is a variable declaration
```

We can declare multiple variables with the same `let` keyword:

```
let name, age, email;
```

After we declared variable, the next step is to store some value in it. This process is called **variable initialization**.

You can do variable initialization at the time of variable creation (when you declare it) or at a later point.

For instance, you might create a variable `student` and assign the value "Josh" to it later.



To put some data into the variable we use the **assignment operator** `=`.

```
let student;  
student = "Josh";
```

You can also assign a value at the time of declaration.

```
let age = 25;
```

If you don't assign a value to a variable when you declare it, its default value will be initialized to `undefined`.

Naming a variable

Rules for naming variables are easy:

1. Names can contain letters (uppercase and lowercase), numbers and the symbols `_` and `$`.

2. The first character of the name can't be a number.

JavaScript allows a **large variety** of characters as variable names, so if you need to use ñ, ö or even π, you can. Make sure your editor saves files with **UTF-8** encoding. All these examples are valid variable names:

```
let a; // the name is a, with var you are only introducing a variable.
let color;
let _private;
let $button;
let getTop10;
let a_large_name;
let thisWayIsCalledCamelCase;
let π;
```



When creating a variable with more than one word, we use the “camelCase” style.

Words or abbreviations in the middle of the phrase begin with a capital letter. The purpose of this practice is to enhance readability.



JavaScript is **case sensitive** so capital letters do make a difference. For example, `color` and `Color` are different variable names.



Although we said you can use any English or non-English word, still there's a bit constraint: there are some **reserved** keywords that **can't be used**, for example: `let`, `class`, `return`, and `function` are reserved.

```
let let = "hello"; // <== error, you can't name variable "let"
```

You can find the full list of **reserved keywords** [here](#).

Changing values

Variables declared using the keyword `let` can be manipulated and their values can be changed throughout the code:

```

let favoriteFood;

favoriteFood = "Steak";
console.log(favoriteFood);


favoriteFood = "Pizza";
console.log(favoriteFood);

// console:
// Steak
// Pizza <== as we can see, the variable changed value from 'Steak' to 'Pizza'

```

In this case, we are just changing the value of the variable but **we can also change its type**.

Type Conversion

 Sidenote: Basic data types in JavaScript, that you are familiar with, are: *string*, *number*, *object*, *array*. In the next lesson we will dig a bit deeper into data types, but for now, just keep these in mind for the following paragraph.

You can reassign values and change the data type of variables in JavaScript.

As we already mentioned, JavaScript is a **dynamically typed language** and that means **new variables are created at runtime**, and the **type of variables is determined at runtime**. The type will get determined automatically while the program is being processed and you can have the same variable at one point as one type and at some other point as some other type.

```


let favoriteFood;

favoriteFood = "Steak";
console.log("Value: ", favoriteFood, " Type: ", typeof favoriteFood);

favoriteFood = 20;
console.log("Value: ", favoriteFood, " Type: ", typeof favoriteFood);

// console:
// Value:  Steak  Type:  string
// Value:  20     Type:  number

```

 Sidenote: `typeof` is JavaScript operator that **returns the type of the variable** we passed to it. You can read more about it [here](#).

As you can see, the type is changed and this is not necessarily a good thing all the time. If you are not careful, this might be a huge headache.

! Don't just try anywhere to change the value of your variables, because there are some other things we have to understand in order to know when and how we can do this. **By changing the value, we can change the type of variable.**

In order to help yourself and others who work or will work on the same code, try to use `const` whenever there is a chance to do so.

Variable declaration with `const`

`const` is used when declaring a variable which value will be constant.

These variables are called **constants** and if we try to change its values we will get the error back:

```
const name = "Ana";
name = "Marina";

// console:
// unknown: "name" is read-only
```

When you are sure a variable will never change, you should declare it with `const` to guarantee and clearly communicate that fact to everyone.

Unlike being able to declare variables and later to assign the value to them using `let` keyword, when using `const` the variable needs to be assigned some value in the same moment when the variable is initialized. This comes natural when you know that variables declared with `const` can't be changed later. (take this conditionally for now, later you'll see that in some sense they can be changed but they can't be reassigned.)

```
let name; // <== we can do this

const price; // <== error 🚨
```

These are basics about variables and we will keep building this knowledge through the next units.

👍 To summarize, for now, you can use `let` or `var` interchangeably when you expect that the variable will change its value. However, use `const` when you're sure it will remain unchanged.

🥁 Important naming rules 🥁

Believe it or not, **naming variables is one of the hardest and most serious things in programming**. It might seem a bit silly, but badly named variables can jeopardize the success and development of your project. Any senior will confirm that just by taking a look at someone's variable names, they already know if they are dealing with someone experienced or with a beginner.

Not all the time you will have a "luck" to start a project from a scratch, but instead you'll be working with someone else's code and you will see that their naming rules impact your speed and code quality a great deal.

💯 You should always think about the next person who will work on your project (or even have to take a look at it), and make sure they don't have to ask you what and why is named the way you named it.

The rules to keep in mind when naming a variable:

- machines process these variables, but humans have to read them and understand them so always use **human-understandable** expressions, like `firstName`, `hasColor`, `pricePreTax`, etc.
- never, ever, ever use `x`, `j`, `a` to name your variables unless it super clear to you and everyone else what is going on in the code
- **name variables as descriptive and as concise as possible** (examples of good naming: `userName`, `creditCardNo`, and examples of bad names: `info`, `value`, etc. which say nothing and mean nothing from someone who was not included in coding process from the very beginning)
- if you're working in a team, and in most cases you will be, this needs to be discussed amongst everybody - terms for naming variables need to be the same for everyone.

Summary

In this introductory lesson, you've had a chance to get familiar with JavaScript as language, to learn a bit about its history and evolution and all the paradigms that JavaScript is part of.

You also learned how to declare a variable and got familiar with `let`, `const` and `var` and learn how to differentiate them on a most basic level.

You also learned that JavaScript is dynamically typed language and that it's possible to change values and types of variables in the runtime (as you code).

And finally, you got familiar with the most important naming rules.

Extra Resources

- [JavaScript](#)
- [MDN](#) This will be your main resource for any JavaScript search. It has tutorials, guides and tools. This is the most thorough JavaScript Documentation.
- [Understanding Javascript Variables](#)