

# JS



## THE ULTIMATE GUIDE TO BECOME A JAVASCRIPT DEVELOPER



**BY - ABEL**



# About this book

**Coding is an incredible ability. It opens the door to a plethora of new possibilities. This is exactly what it did for me and many others I know!**

**This is not the type of book where you have to go over theoretical material every time, instead, it is jam-packed with the resources you will need on your way to becoming a successful Javascript developer.**

**I don't claim to know everything, therefore I'm continually learning, and this book is in "continuous growth" mode. As I experiment and learn new things, I'll continue to add to the book.**

## How to read the book?

**This book is written from a beginner's perspective, so use it as a road map for learning Javascript.**

**You can start reading any chapter you like because this book features distinct chapters that allow you to read what you want without having to read it in order. Feel free to jump to any part that interests you.**



***DEDICATED TO  
ALL NEW DEVELOPERS***

# Let's Start

## What is JavaScript ?

- JavaScript is an interpreted language
- It means it doesn't need a compiler
- It executes instructions directly without compiling
- It is platform independence, dynamic typing
- The source code is evaluated JUST before executing it
- It is open-source and cross-platform compatible
- It is created by NetScape
- It has object-oriented capabilities

# Who created JavaScript ?

**Brendan Eich** when working at NetScape  
and It was created in 10 days


## Why do you love JavaScript?

- It is easy to start using
- JavaScript can be used on any platform
- It performs well on every platform
- You can build web, IOT, mobile apps using JavaScript

- It can be used on the Frontend, Backend, and also in the databases like MongoDB
- It is dynamic in nature ex: objects and arrays can be of mixed types

# Your first "hello world" program

Write the below HTML code in index.html file and open it in browser



```
<!DOCTYPE html>
<html>
  <body>
    <h1>My First Web Page</h1>
    <script>
      console.log("Hello World");
    </script>
  </body>
</html>
```

- JavaScript code is written in between the script tag in the above code.
- When the page loads the browser will run the code between the script tag.
- alert() function will be called which will create a model with hello world text on it.

**Congratulation! You just wrote your first JavaScript program**

# How does JavaScript work?

JavaScript is a scripting language. This means it is used to automate processes that users would otherwise need to execute on their own, and step-by-step.

Also, without JS, any changes on web pages you visit would require either manually reloading the page, or navigating a series of static menus to get to the content you're after.

JavaScript does the heavy lifting by telling computer programs like web browsers or web applications to do something specific.

This is where dynamic features come in, for instance, to “tell” an image to animate itself, a photo to move through a slideshow, etc.

As JavaScript is such an integral part of web functionality, all web browsers come with built-in engines that can render JS meaning that you can type JS commands directly into an HTML document and the web browser will understand them.

JavaScript is typically embedded into a web page or included in a .JS file. Being a client-side language, the web browser downloads it on your computers for processing; no need for any additional programs or compilers.



# JavaScript Terminology

**JavaScript is an interpreted programming language that conforms to the ECMAScript specification.**

**JavaScript is high-level, often just-in-time compiled, and multi-paradigm.**

**It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.**

- Semicolons – Let's start with a controversial aspect. You should use semicolons to terminate every line of your JS programme, but JavaScript interpreters usually do it for you. Thus some developers use them, some others don't
- Values – It indicates values such as hello and 5 where string and number are the types of these values. JS has several types with their unique characteristics. To have a reference for a value, you can then assign it to a variable

- Variables – A variable is a value you assign to an identifier. You can name a variable and then get quick access to any value you stored in it just by searching for the variable's name. To use a variable, first you must declare it either with a `const` or a `let`, the difference being that the former cannot be reassigned a new value, while the latter can. To avoid bugs, experienced JS developers prefer to use `const` more than `let` – unless they know they'll need to reassign a value to that variable. Another kind of variable is `var`, but it's becoming less and less utilised.
- Types – In JavaScript there are two main kinds of types: primitive types (numbers, strings, booleans, symbols) and object types (a string, a number, a boolean, null or undefined). Object types also have properties and methods. You can assign a value with some type to a variable and later reassign the variable to host a different value. Remember, JS Variables are untyped
- Expressions – It's a single JS unit that the JavaScript engine can evaluate, and return a value. You can find very simple expressions, called primary expressions, less simple ones like arithmetic expressions, string expressions, logical expressions, and even more complex ones such as those involving objects, functions, and arrays

- Operators – With them, you can combine two simple expressions to form a more complex one. Operators are classified according to the operands they work with. Some work with 1 operand, most work with 2 operands, and just one operator works with 3 operands. Here's a list of operators:
  - Addition (+)
  - Subtraction (-)
  - Division (/)
  - Remainder (%)
  - Multiplication (\*)
  - Exponentiation (\*\*)
  - Comparison (true or false)
  - Conditionals (if – else)
  -
- Arrays – They are a collection of elements. Arrays are objects that can hold any value, even of different types, and are multi-dimensional (i.e. you can put an array into another array), which makes them ideal for creating a matrix
- Strings – They are a sequence of characters that can be enclosed in quotes or double quotes (literals). Two strings can be joined using the + operator. You can also define strings using template literals, defined inside backticks, to make multiline strings much simpler and also provide an easy way to interpolate variables and expressions into strings
- Loops – They are one of JavaScript's control structures. Loops can automate and repeat a block of code as many times you want. There are many ways to use loops, but the most common ones are:
  - while loops
  - for loops
  - for of loops

- **Functions** – A function is a self-contained block of code, and a core part of JavaScript. Functions can be defined inside other functions, but the nested function cannot be called from the outside of the enclosing function
- **Arrow functions** – Similar to Functions, they allow you to write functions with a shorter syntax. But these functions are anonymous so you must assign them to a variable. Like Functions, they can have default values for parameters, only return one value, and contain other arrow functions, or even regular functions. The big difference is when they are used as object methods
- **Objects** – An Object is any non-primitive value (a string, a number, a boolean, a symbol, null, or undefined). They are passed by reference. Arrays and functions are objects in practical terms. Objects have *Properties, Methods, Classes and Inheritance*

- Asynchronous Programming & Callbacks – JavaScript code is usually run synchronously. But if your program cannot wait for a line of code to execute before another one – as it commonly occurs when working with the file system, the network, events, or the DOM in the browser – then JavaScript uses callbacks. This function is executed asynchronously as it passes a function to another function call that is called when the function finishes processing.
- Promises – Promises offer an alternative way to satisfy your asynchronous coding needs. Their primary intent is to avoid the so-called callback hell; if we need to use the result of a function in the rest of your code, all our code must be nested inside the callback. Multiply that by 2-3 callbacks and you end up messing up things.
- Async and Await – An async function typically returns a promise as async functions are abstractions of promises. The async/await duo helps you to return a cleaner code as well as a simpler mindframe to work with asynchronous code.
- Variable scope – Scope is the set of variables that is only visible to a portion of your program. Scopes in JS are divided into global scope, block scope and function scope. A variable defined outside of a function – or block – has a global scope making it available in every part of your program

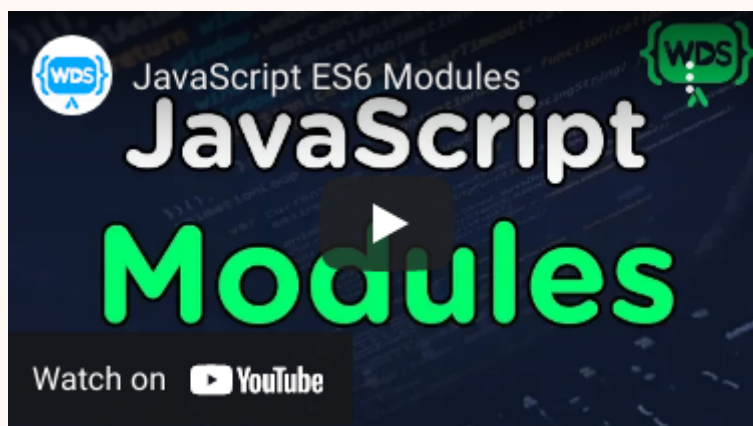
# Resources to Learn through Videos

## 1. JavascriptTutorial for Beginners [From 0 to ES6+] - Full Course (Scrimba)

This video teaches you the basics of JavaScript, the most popular programming language in the world.



## 2. With ES6 JavaScript changed from a programming language that many people dreaded using to one of the most popular and loved languages.



### 3. JavaScript Promises

ES6 came with many new features, but one of the best features was the official introduction of Promises. Promises allow you to write clean non-callback-centric code without ever having to worry about callback hell. Even if you never write your own promise, knowing how they work is incredibly important, since many newer parts of the JavaScript API use promises instead of callbacks. Checkout the full video to learn how to define and use promises.



### JavaScript Promises In 10 Minutes

### 4. Asynchronous JavaScript Course (Async/Await, Promises, Callbacks)

Learn how to use Async/Await, Promises, and Callbacks in JavaScript.





## 5. 30 Day Vanilla JS Coding Challenge

**Build 30 things in 30 days with 30 tutorials**

[https://www.youtube.com/watch?v=VuN8qwZoego&list=PLu8EoSxDXHP6CGK4YVJhL\\_VWetA865GOH](https://www.youtube.com/watch?v=VuN8qwZoego&list=PLu8EoSxDXHP6CGK4YVJhL_VWetA865GOH)



Scan this for accessing Javascript 30





**That's all there is to it!**

**If you carefully follow this instruction and watch all of the lessons, you should have enough confidence to begin constructing projects on your own by now. You might also begin to study Javascript libraries such as React, Angular, or Vue JS.**

**This is simply a beginner's tutorial to get you started, and in the coming days, I'll be posting a lot of advanced professional materials to help you understand Javascript and Solidity and become a Web 3 Developer.**

**THANKYOU**