The Complete JavaScript Guide

17+ topics, 5+ cheatsheets, 3+ hours of work

To start using JavaScript, you only need this blog.

What is JavaScript

JavaScript is a multi-paradigm, dynamic, high-level, loosely typed language.

We use JavaScript to create

• websites

• Mobile applications

• web applications

• server-side applications using Node.js

- smartwatch applications

Let now cover all the important JavaScript concepts, one-by-one

1. Variable

A variable is a value assigned to an identifier, so you can reference and use it later in the program.

We use the following keywords to declare variables:

`let`

`const`

`var`

2. Types

The type defines the data type of a particular variable or method.

Two main types of types in JavaScript:

• Primitive types

• object types

Primitive types are those which are either numbers or strings or booleans.

Object types are those which are either numbers or strings or booleans(if defined with the new keyword)

3. Operators

In JavaScript, an operator is a special symbol used to perform operations on operands (values and variables).

The operators present in JavaScript are:

• Addition +

• Subtraction -

• Multiplication *

• Division /

• modulus %

• Exponential **

Comparison operators

<, >, <=, >=, ==, ===, !==

4. Conditionals

Conditional statements control behavior in JavaScript and determine whether or not pieces of code can run.

```
if(true){ condition}
```

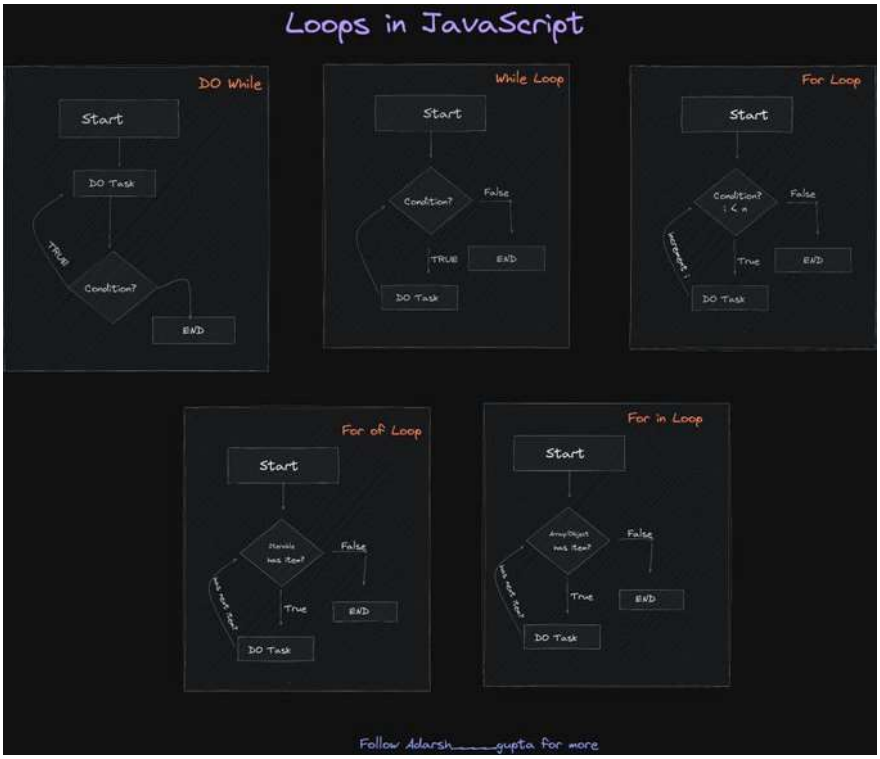
Loops

```
While loop: while(true){}

For Loop: for(s1,s2,s3){//statements}

Do while: do{}while(true)
```

5. Complete loop Cheatsheet


Loops in JavaScript
DO While
Start
DO Task
TRUE
Condition?
END
While Loop
Start
Condition?
False
TRUE
END
DO Task
For Loop
Start
Condition?
i < n
False
INCREMENT i
True
END
DO Task
For of Loop
Start
Variable
has item?
False
True
END
DO Task
For in Loop
Start
Array/Object
has item?
False
True
END
DO Task
Follow Adarsh_____gupta for more

6. Arrays

An array is a collection of similar data elements stored at contiguous memory locations

```
const array=[1,2,3]
```

Array Operations

```
push() : Add to array

pop() : remove from array

concat() : join 2 arrays
```

7. Hoisting

When developers are unclear about the concept of hoisting in JavaScript, they frequently encounter unexpected outcomes. Before the execution of the code, the interpreter appears to move the declaration of functions, variables, or classes to the top of their scope.

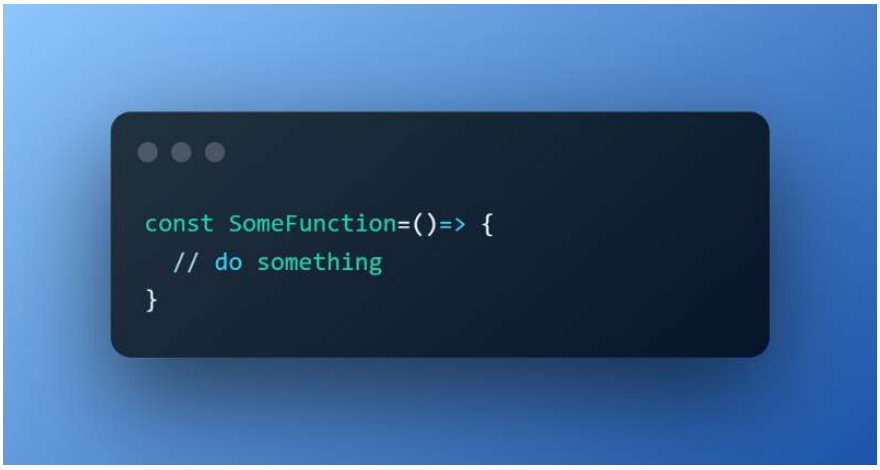
8. Functions

lines of code for doing a specific task.

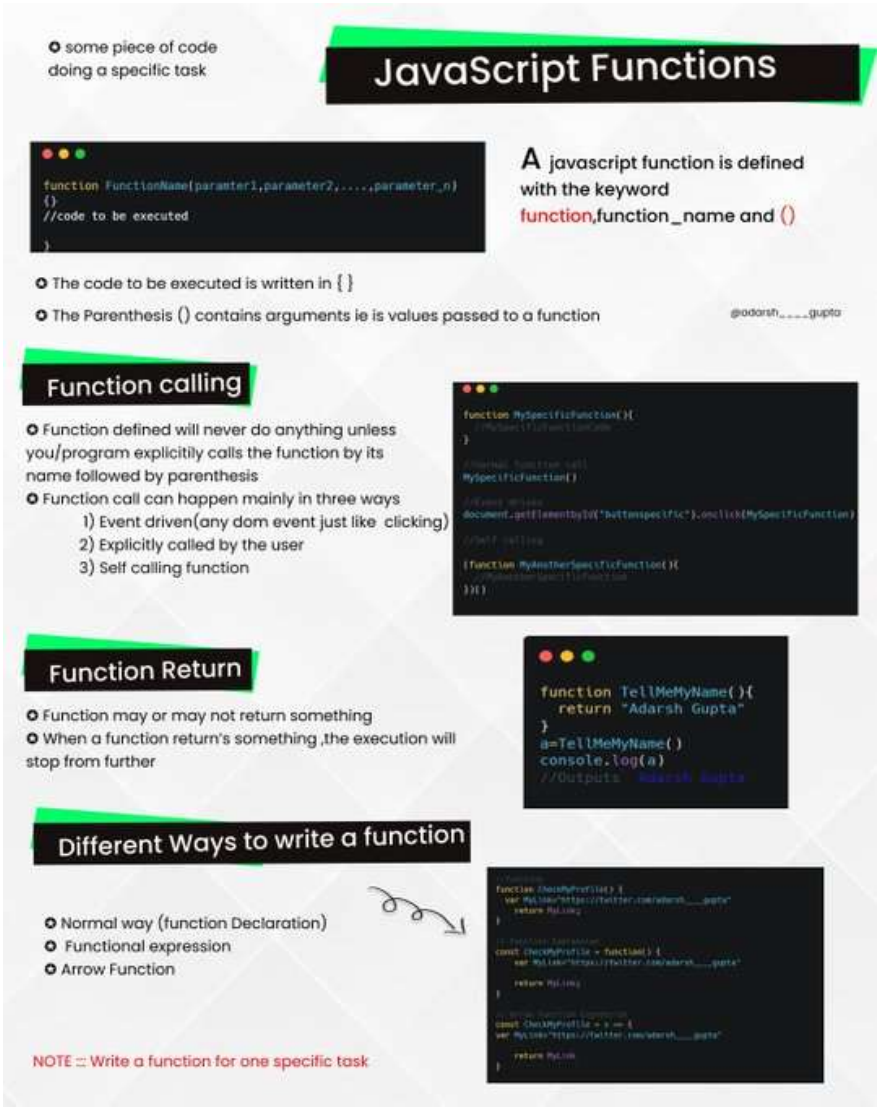
A function in JavaScript is similar to a procedure—a set of statements that perform a task or calculate a value—but for a procedure to qualify as a function, it should take some input and return an output where there is some obvious relationship between the input and the output(according to MDN docs).

9. Arrow functions

They allow you to write functions with a shorter syntax.



10. Complete functions Cheatsheet



11. Scope

Scope defines whether you can access or reference a particular value or expression. We are unable to use a declared variable if it is not included in the current scope. This idea is crucial to understand because it makes it easier to separate logic in your code.

In JavaScript, we have 3 types of scopes:

Global scope: Variables and expressions can be referred to anywhere in a global scope. This is the default scope.

Local scope: variables and expressions can be referenced only within the boundary.

12. Objects

Any value that's not of a primitive type is always passed by reference.

13. Classes

Class methods are created with the same syntax as object methods. Use the keyword class to create a class.

Classes are a template for creating objects.

```
class Person {
  constructor(name) {
    this.name = name
  }

  breathe() {
    return ' I am ' + this.name + 'and I'm breathing'
  }
}
```

14. Callbacks

A callback is a function that is passed as an argument to another function, and its execution is delayed until that function to which it is passed is executed.

```
COPY
//callback function with parameters and return value

function greeting(parameter) {
  console.log(parameter);
}
function callBackFunction(callback) {
  callback("Hello World");
   console.log("I am in a callback function");

}

callBackFunction(greeting);
```

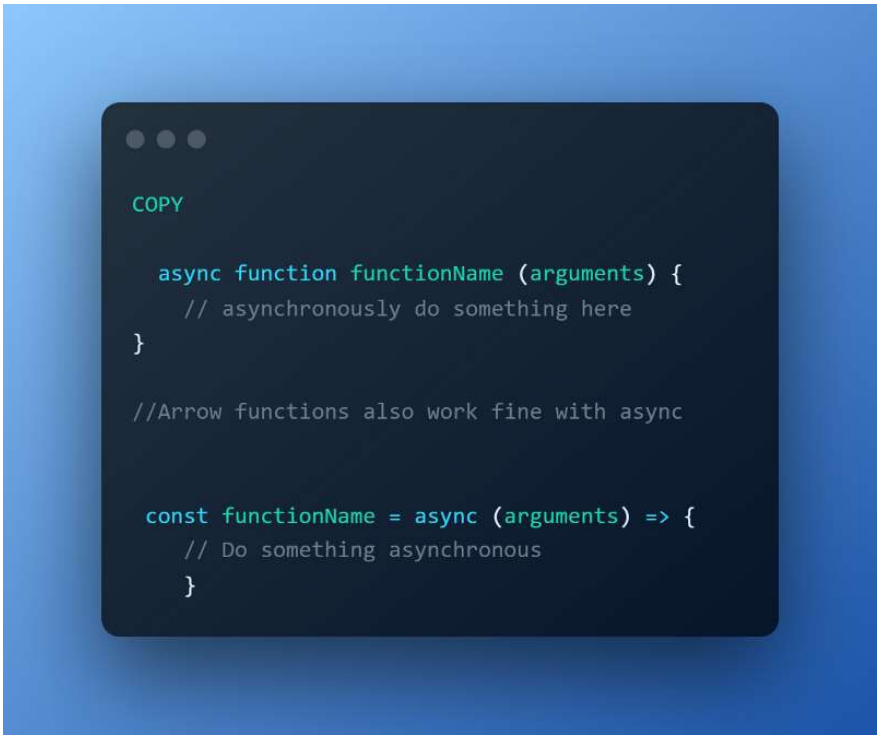
15. Promises

A promise is an object that has the potential to produce only one value in the future: either a resolved value or an explanation for why it cannot be resolved (such as a network error).

There are three possible states for a promise: fulfilled, rejected or pending

16. Asynchronous JavaScript

Asynchronous JavaScript has never been easy. We have used callbacks for a while. Then, we made promises. We use asynchronous functions most of the time now.



17. Closure

It is a feature in JavaScript where an inner function has access to the outer function's variables

The inner function can access the variables defined in its scope, the scope of its parent functions, or even its grandparent functions and the global variables.

```
function grandparent() {
    let car="BMW"
    var parent = function () {
        let house="4BHK"
        var child = function () {
            return ("The child gets " + car + " and " +
house);
        };
        return child;
    };
    return parent;
}

console.log( grandparent()()()) //The Child gets BMW and
4BHK house
```

Wrapping it up

It's a strange language, JavaScript. But when you look closer, you typically understand why things operate that way.

I sincerely hope that this list will help you understand some of the crucial JavaScript concepts that you should know, and if you are aware of any additional concepts that are noteworthy, please mention them in the comments section :)

This guide took several hours to create, and several resources have been referenced. If you find this Guide useful, Share and spread the word

Follow me, Adarsh gupta on Medium as well as on Twitter (@Adarsh____gupta).

If you wish, you can support me by buying me a Chai.