Index.

Why you should Learn JavaScript?

It Works in The Browser

Like most languages, you don't need to setup anything. You can run your code without any Environment.

Easy to Learn

A very beginner friendly language in which you don't need to learn deal with complexities.

Versatile Programming Language

From front-end to back-end, JavaScript can be used for almost anything. There's nothing you can't do with JS.

Big Community Support

Doesn't matter what error you face while learning. Just google it, and you'll see tons of solution.

Let, Const, Var

Let

The let statement declares a block-scoped local variable, optionally initializing it to a value.

var

The var statement declares a function-scoped or globally-scoped variable, optionally initializing it to a value.

const

Constants are block-scoped, much like variable is declared using the let keyword. The value of a constant can't be changed through reassignment, and it can't be redeclared.

Cheatsheet. 😍

Keyword	Function vs Block-Scope	Redefinable
var	function-scope	✓
let	block-scope	✓
const	block-scope	✗

Operations

Definition

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

```
2 + 3; // 5
```

+ is an operator that performs addition, and 2 and 3 are operands.

JavaScript Operator Types

- Assignment Operators.
- Arithmetic Operators.
- Comparison Operators.
- Logical Operators.
- Bitwise Operators.
- String Operators.
- Other Operators.

Data Types

Data types can be confusing sometimes. Let's make it clear because its super important.

Definition

```
const x = 5;
const y = "Hello";
```

- 5 is an integer data.
- "Hello" is a string data

String	represents textual data	`'hello world!'`
Number	an integer or a floating-point number	`3e-2` `3.234` `3`
BigInt	an integer with arbitrary precision	`900719925124740999n`
Boolean	Any of two values: true or false	`true` `false`
undefined	a data type whose variable is not initialized	`let a;`
null	denotes a null value	`let a = null;`

Symbol	data type whose instances are unique and immutable	`let value = Symbol('hello');`

Object	key-value pairs of collection of data	`let student = { };`

Strings

JavaScript String

JavaScript string is a primitive data type that is used to work with texts. For example,

```
const name = 'John';
```

Creating JavaScript String.

Strings are created by surrounding them with quotes. Here's how we do it:

Single quotes: 'Hello'
Double quotes: "Hello"
Backticks: `Hello`

- Single & Double quotes are practically the same. Use Either of them.
- We use backticks when we need to include variables or expression in a string.

Events

JavaScript Events

The change in the state of an object is known as an Event. This process of reacting over the events is called Event Handling. Thus, JS handles the HTML events via Event Handlers.

onclick — The event occurs when the user clicks on an element.
oncontextmenu — User right-clicks on an element to open a context menu.
ondblclick — The user double-clicks on an element.
onmousedown — User presses a mouse button over an element.
onmouseenter — The pointer moves onto an element.
onmouseleave — Pointer moves out of an element.
onmousemove — The pointer is moving while it is over an element.

Event Listener/ Event Handler

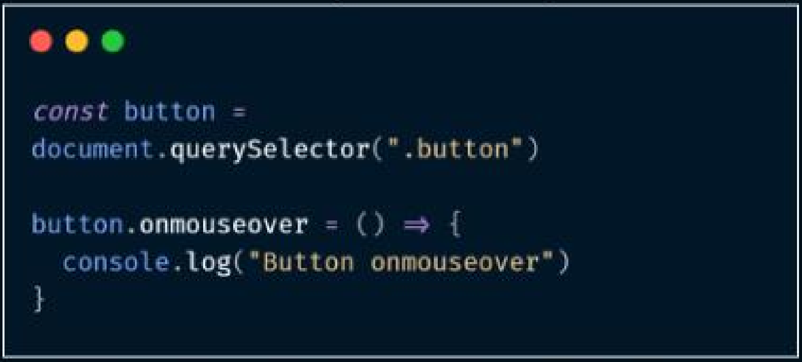
There are two ways you can handle an event in JS.

- Event Listeners.
- Event Handlers.

Event Handlers

• To use event handlers, use one of the EventHandler properties of an object.

• Here's an example of using one onmouseover.



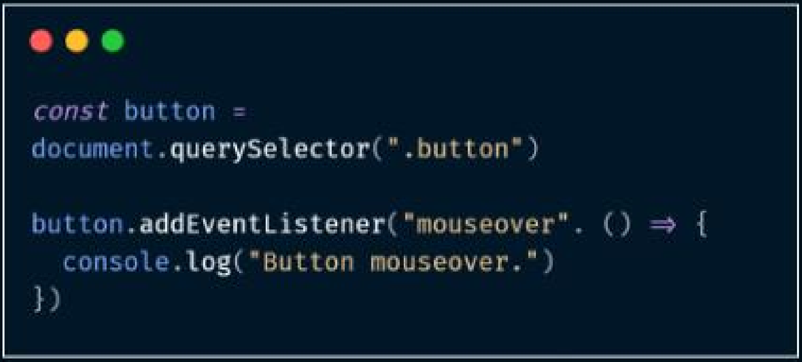
```javascript
const button =
document.querySelector(".button")

button.onmouseover = () => {
  console.log("Button onmouseover")
}
```

• onmouseover - triggers when the mouse pointer is moved onto the button.

Event Listener

• An event listener is something you assign to an object.

• As the name suggests, the event listener listens for events and gets triggered when an event occurs.



```javascript
const button =
document.querySelector(".button")

button.addEventListener("mouseover", () => {
  console.log("Button mouseover.")
})
```

• mouseover - Hovering on the button triggers a "mouseover" event, then it runs the block of code.

Functions

JavaScript Functions

JavaScript provides functions similar to most of the scripting and programming languages. In JavaScript, a function allows you to define a block of code, give it a name and then execute it as many times as you want.

```
//defining a function
function <function-name>()
{
    // code to be executed
};

//calling a function
<function-name>();
```

Example

```
function ShowMessage() {
    alert("Hello World!");
}
ShowMessage();
```

Objects

JavaScript Objects

JavaScript object is a non-primitive data-type that allows you to store multiple collections of data.

```
// object
const student = {
    firstName: 'ram',
    class: 10
};
```

Here, student is an object that stores values such as strings and numbers.

JavaScript Object Declaration

```
const object_name = {
   key1: value1,
   key2: value2
}
```

Here, an object object_name is defined. Each member of an object is a key: value pair separated by commas and enclosed in curly braces {}.

Arrays

JavaScript Arrays.

An array is an object that can store multiple elements. For example,

```
const myArray = ['hello', 'world', 'welcome'];
```

Array Example.

```
// empty array
const myList = [ ];
// array of numbers
const numberArray = [ 2, 4, 6, 8];
// array of strings
const stringArray = [ 'eat', 'work', 'sleep'];
// array with mixed data types
const newData = ['work', 'exercise', 1, true];
```

getter

what is getter -

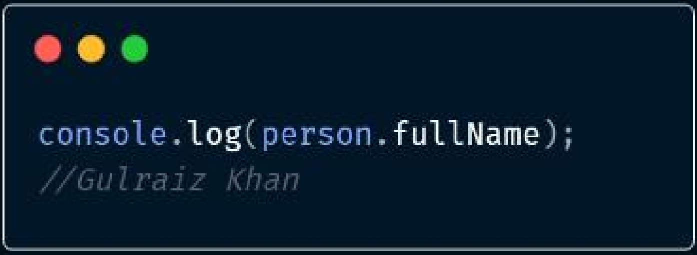
we use getter to access the properties.

In this case, we have **firstName** & **lastName,** but what if we want to access full name. Here's how we will do it.

```
const person = {
  firstName = 'Gulraiz',
  lastName = 'Khan',
  fullName () {
    return `${person.firstName}
${person.lastName}`
  }
}
```

in **fullName()**, we're using a template literal.

Output -

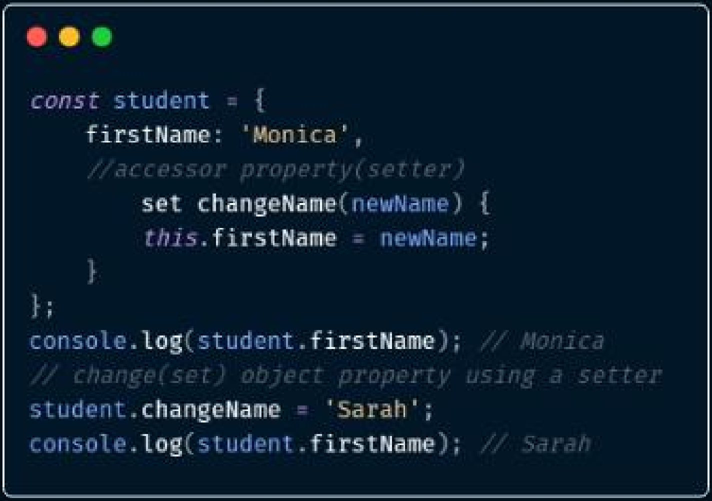


```
console.log(person.fullName);
//Gulraiz Khan
```

setter

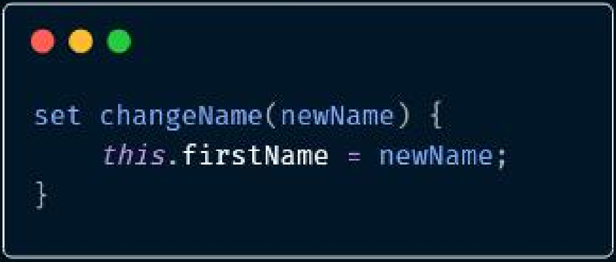
what is setter -

we use setter to change (mutate) the property.



```
const student = {
    firstName: 'Monica',
    //accessor property(setter)
        set changeName(newName) {
        this.firstName = newName;
    }
};
console.log(student.firstName); // Monica
// change(set) object property using a setter
student.changeName = 'Sarah';
console.log(student.firstName); // Sarah
```

In the above example, the setter method is used to change the value of an object.



```
set changeName(newName) {
    this.firstName = newName;
}
```

To create a setter method, the set keyword is used.

for loop

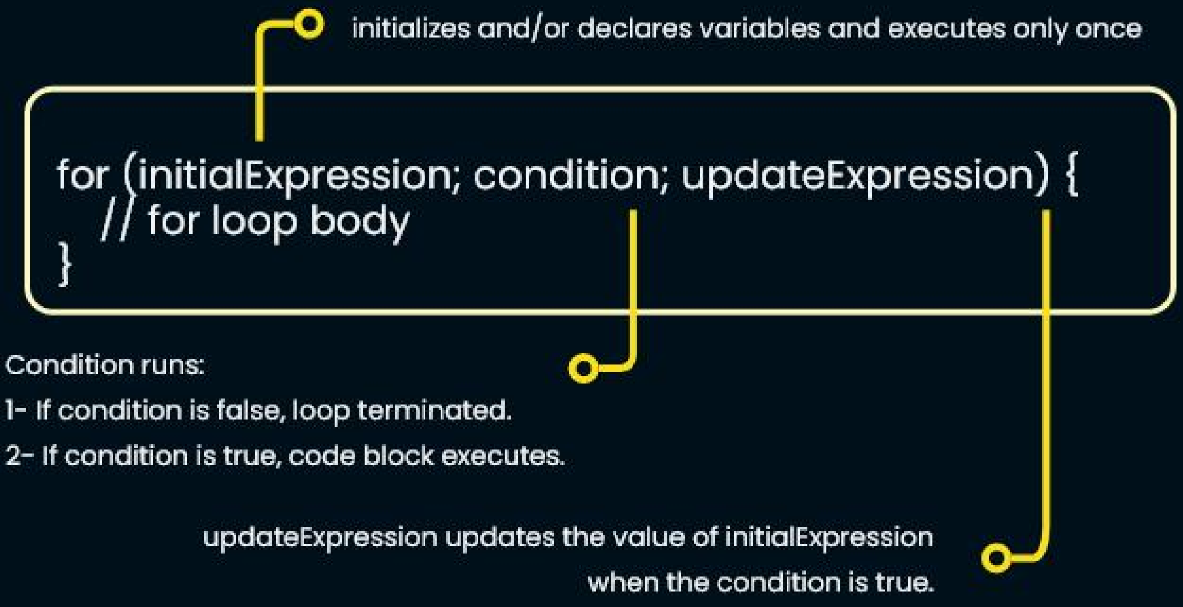
For Loop

Loops are used to repeat a block of code.

Example

If you want to show something 1000 times, you can use a loop.

Syntax



initializes and/or declares variables and executes only once

```
for (initialExpression; condition; updateExpression) {
   // for loop body
}
```

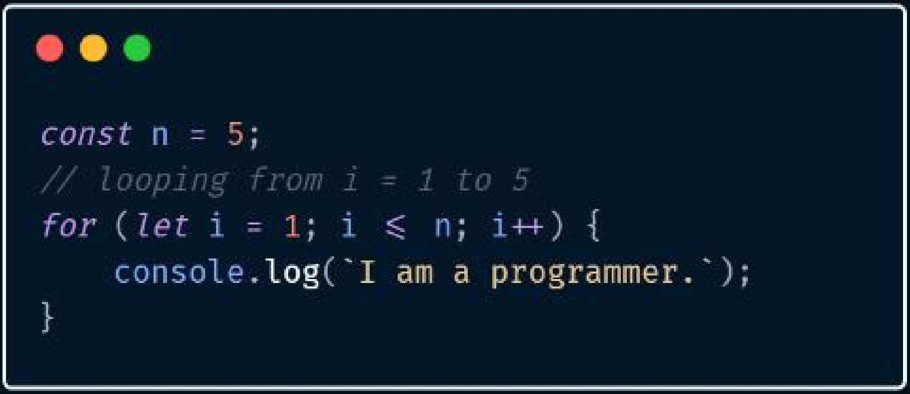
Condition runs:

1- If condition is false, loop terminated.

2- If condition is true, code block executes.

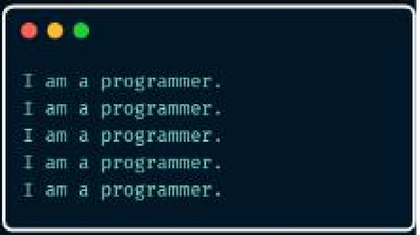
updateExpression updates the value of initialExpression when the condition is true.

Example -



```
const n = 5;
// looping from i = 1 to 5
for (let i = 1; i <= n; i++) {
    console.log(`I am a programmer.`);
}
```

Output -



```
I am a programmer.
I am a programmer.
I am a programmer.
I am a programmer.
I am a programmer.
```

for...in

Syntax

```
for (key in object) {
    // body of for...in
}
```

In each iteration of the loop, a key is assigned to the key variable. The loop continues for all object properties.

Once you get keys, you can easily find their corresponding values.

Example -

```
const student = {
    name: 'Monica',
    class: 7,
    age: 12
}
// using for...in
for ( let key in student ) {
    // display the properties
    console.log(`${key} => ${student[key]}`);
}
```

Output -

```
name => Monica
class => 7
age => 12
```

Code Explanation -

In the above program, the for...in loop is used to iterate over the student object and print all its properties.

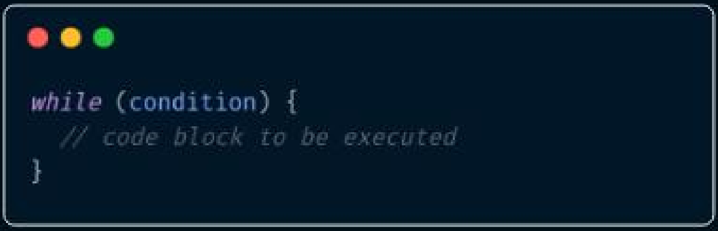
1- The object key is assigned to the variable key.
2- student[key] is used to access the value of key.

while loop

While Loop

The while loop loops through a block of code as long as a specified condition is true.

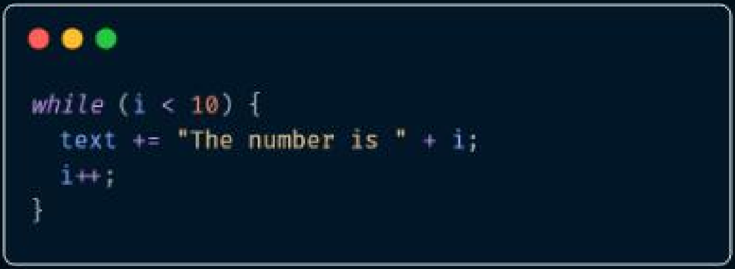
Syntax



```
while (condition) {
  // code block to be executed
}
```

Example -

the code in the loop will run, over and over again, as long as a variable (i) is less than 10:



```
while (i < 10) {
  text += "The number is " + i;
  i++;
}
```

do while loop

Do while statement creates a loop that executes a specified statement until the test condition evaluates to false.

Syntax

```
do {
    // body of loop
} while(condition)
```

How it works -

>> The body of loop is executed.

>> If the condition is true, the body of the loop inside the do statement is executed again.

>> The condition is evaluated again.

>> If the condition evaluates to true, the body inside do is executed again.

>> The process continues until the condition evaluates to false. Then the loop stops.

do...while loop is similar to the while loop. The only difference is that in do...while loop, the body of loop is executed at least once.

Type Conversion

Break Statement

Type conversion is the process of converting data of one type to another.

Example -

Converting String data to Number.

Types

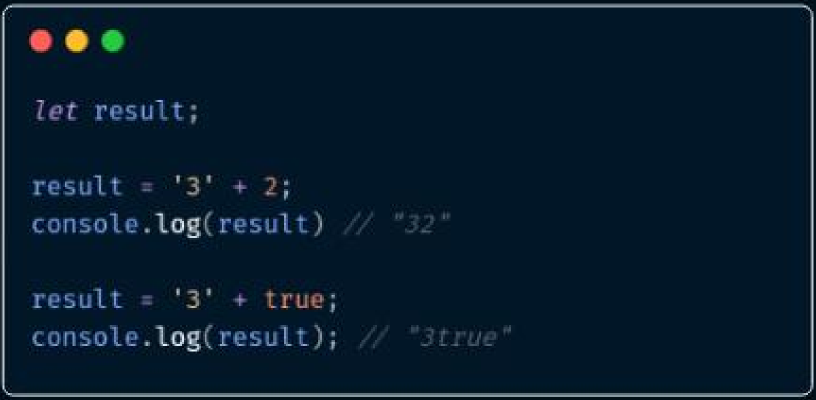
>> **Implicit Conversion** - automatic type conversion

>> **Explicit Conversion** - manual type conversion

// Implicit Conversion

In certain situations, JavaScript automatically converts one data type to another (to the right type).

Example

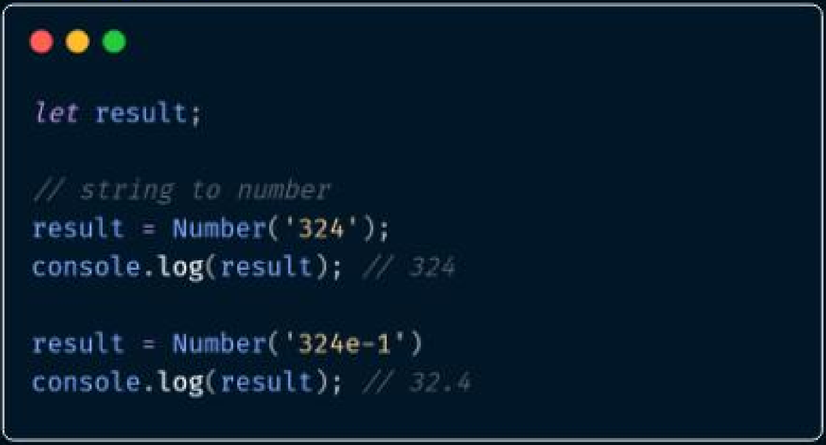


```javascript
let result;

result = '3' + 2;
console.log(result) // "32"

result = '3' + true;
console.log(result); // "3true"
```

// Explicit Conversion

The type conversion that you do manually is known as explicit type conversion.

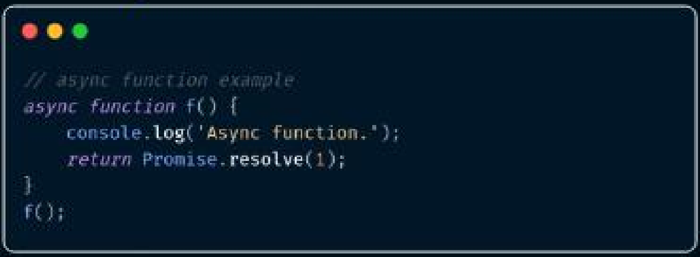
Example



```javascript
let result;

// string to number
result = Number('324');
console.log(result); // 324

result = Number('324e-1')
console.log(result); // 32.4
```

Here,
>> name - name of the function
>> parameters - parameters that are passed to the function

Example



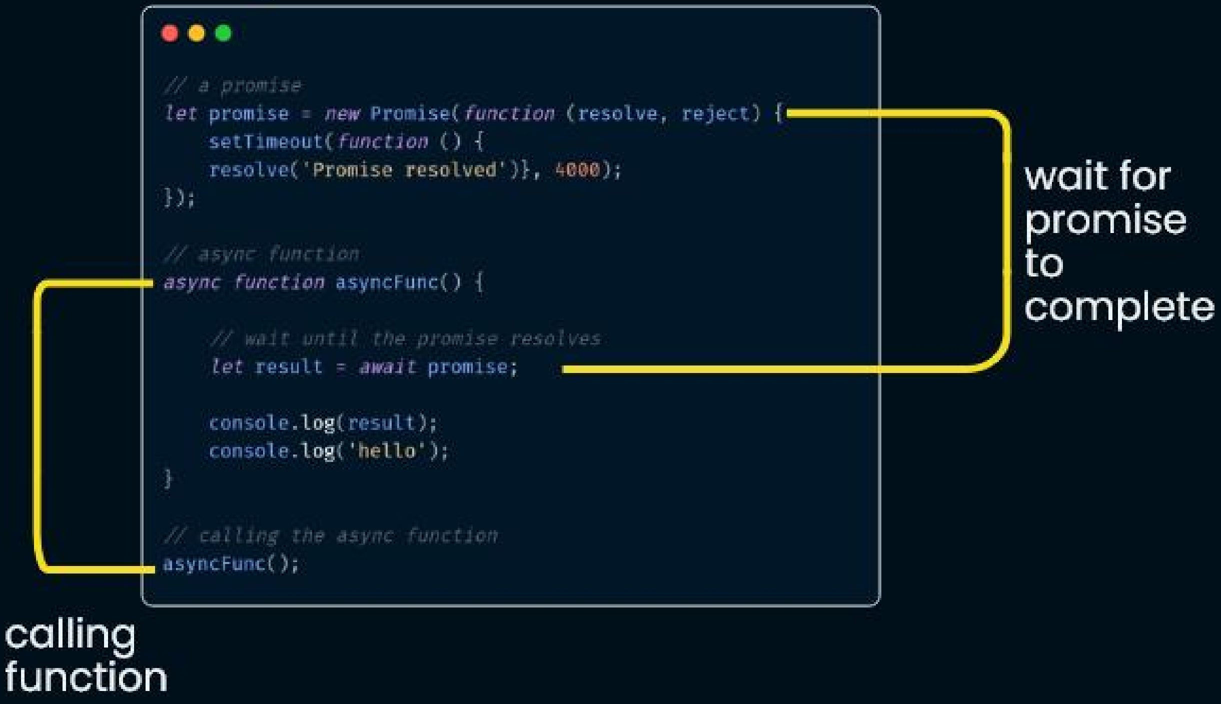
```javascript
// async function example
async function f() {
    console.log('Async function.');
    return Promise.resolve(1);
}
f();
```

Output

async function.

The **await** keyword is used **inside** the async function to **wait** for the asynchronous operation.

Example



```javascript
// a promise
let promise = new Promise(function (resolve, reject) {
    setTimeout(function () {
    resolve('Promise resolved')}, 4000);
});

// async function
async function asyncFunc() {

    // wait until the promise resolves
    let result = await promise;

    console.log(result);
    console.log('hello');
}

// calling the async function
asyncFunc();
```

Note: You can use await only inside of async functions.