Features Explanation

You must learn!

Hasibul Islam
@devhasibulsilam

Dynamic Typing

JavaScript is dynamically typed, meaning you don't have to explicitly specify the data type of a variable. The type is determined at runtime.

```
let myVariable = 10; // Number
myVariable = "Hello"; // String
```

Template Literals

Template literals allow you to embed expressions inside string literals, making it easier to concatenate variables into strings.

```
let name = 'John';
let greeting = `Hello, ${name}!`;
```

Hasibul Islam
@devhasibulsilam

Prototypal Inheritance

JavaScript uses prototypal inheritance, allowing objects to inherit properties and methods from other objects.

```
// Parent object
let animal = {
  sound: 'Animal Sound',
  makeSound: function() {
    console.log(this.sound);
  }
};

// Child object inherits from the parent
let dog = Object.create(animal);
dog.sound = 'Bark';
dog.makeSound(); // Outputs: Bark
```

Hasibul Islam
@devhasibulsilam

Closures

JavaScript supports closures, which allow functions to retain access to variables from their containing scope even after the scope has finished executing.

```
function outerFunction() {
  let outerVariable = 'I am from outer function';

  function innerFunction() {
    console.log(outerVariable);
  }

  return innerFunction;
}

let closure = outerFunction();
closure(); // Outputs: I am from outer function
```

Hasibul Islam
@devhasibulsilam

Asynchronous Programming (Callbacks, Promises, Async/Await)

JavaScript enables asynchronous programming using callbacks, promises, and async/await to handle tasks such as fetching data from a server without blocking the main thread.

```
function fetchData() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve('Data fetched successfully!');
    }, 2000);
  });
}

fetchData().then(data => {
  console.log(data);
});
```

Hasibul Islam
@devhasibulsilam

Arrow Functions

Arrow functions provide a concise syntax for writing functions in JavaScript.

```
// Regular function
function add(a, b) {
  return a + b;
}

// Arrow function
let addArrow = (a, b) => a + b;
```

Hasibul Islam
@devhasibulsilam

Modules

JavaScript supports modular programming using the import and export keywords, allowing you to organize code into separate files.

```
// math.js
export const sum = (a, b) => a + b;

// app.js
import { sum } from './math';
console.log(sum(5, 10)); // Outputs: 15
```

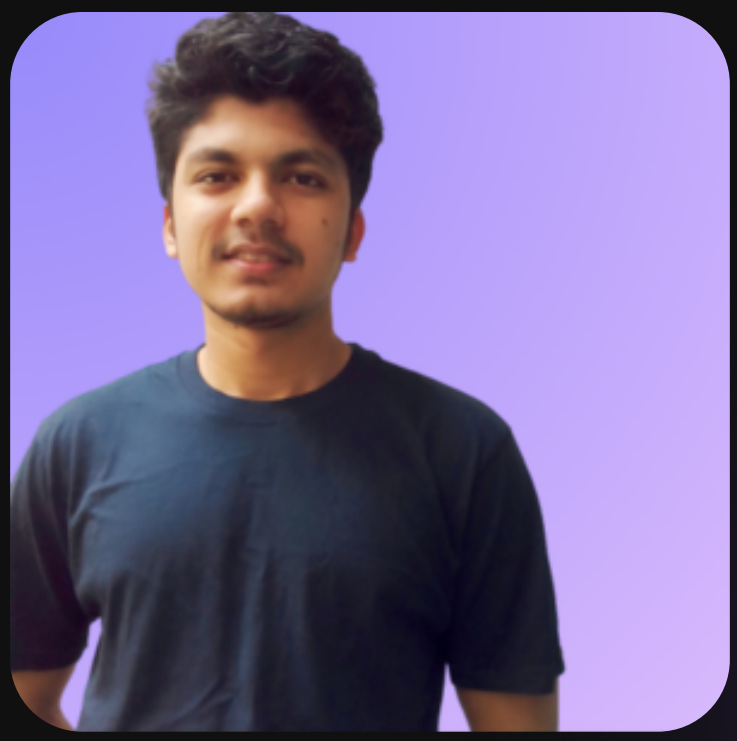
Hasibul Islam
@devhasibulsilam

ES6 Destructuring

Destructuring allows you to extract values from arrays or objects and assign them to variables in a more concise way.

```
// Array destructuring
let [first, second] = [1, 2, 3];
console.log(first, second); // Outputs: 1 2

// Object destructuring
let person = { name: 'Alice', age: 30 };
let { name, age } = person;
```



Hasibul Islam

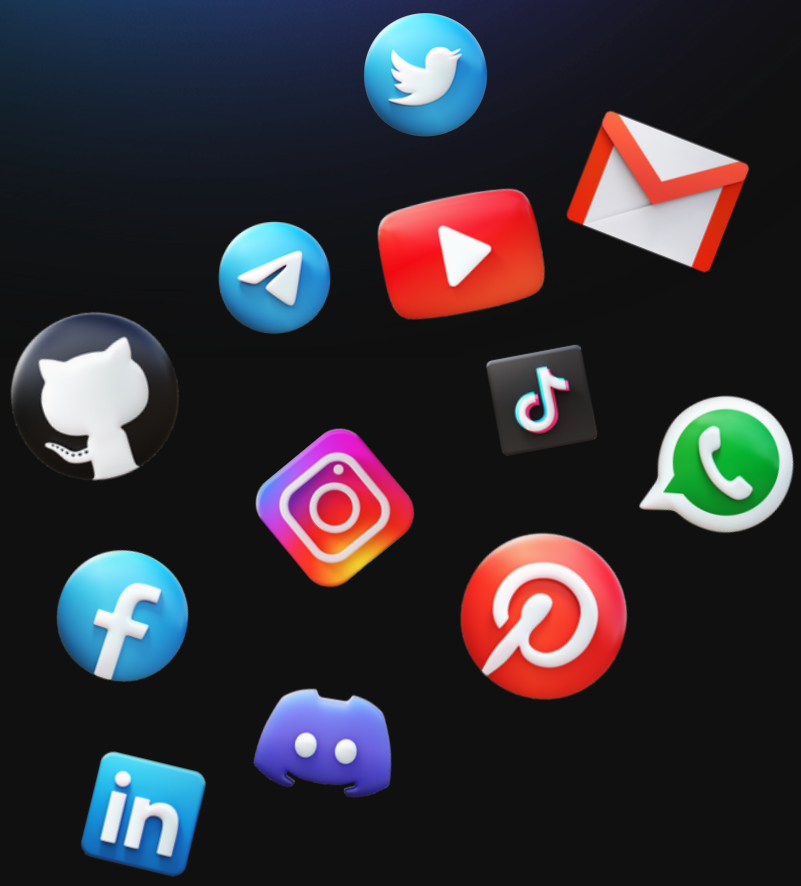
@devhasibulislam

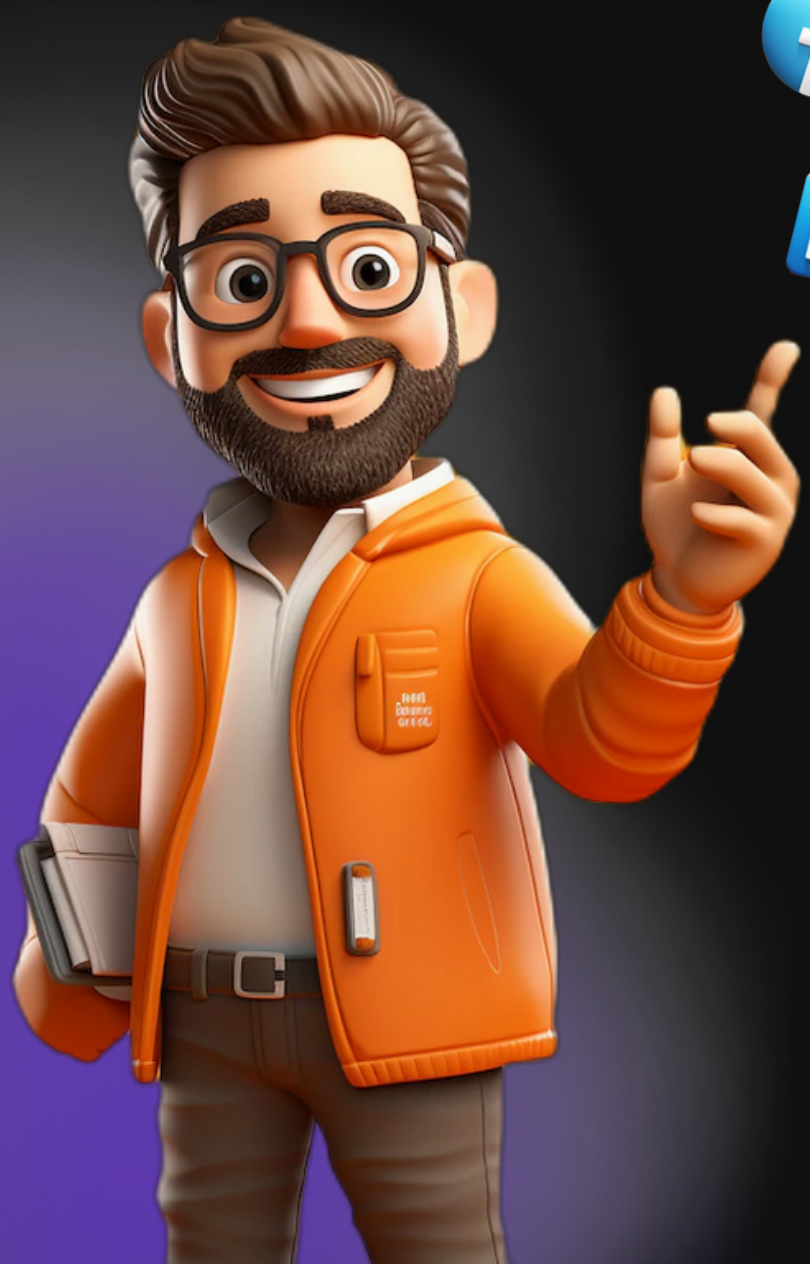
Did you find it useful?

Share your thoughts.

Type on social

devhasibulislam





Follow me