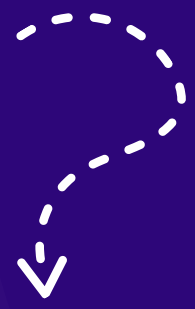
SHARPING

THE BRAIN



PART - 1

JAVASCRIPT

let and const

ES6 introduced block-scoped variables using let and const. let allows you to declare variables that can be reassigned, while const is used for variables that should not be reassigned.

```
let x = 10;
const y = 20;
```

Arrow Functions

ES6 introduced block-scoped variables using let and const. let allows you to declare variables that can be reassigned, while const is used for variables that should not be reassigned.

```
const add = (a, b) => a + b;
```

Template Literals

Template literals allow you to create strings with embedded expressions using backticks (`).

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

Destructuring

Destructuring allows you to extract values from objects and arrays more easily.

```
const { firstName, lastName } = person;
const [first, second] = numbers;
```

Default Parameters

You can set default values for function parameters.

```
function greet(name = 'Guest') {
    console.log(`Hello, ${name}!`);
}
```

Spread and Rest Operators

The spread (...) and rest (...) operators simplify working with arrays and object properties.

```
const arr = [1, 2, 3];
const newArr = [...arr, 4, 5];
function sum(...args) {
    return args.reduce((total, num) =>
total + num, 0);
}
```

Classes

ES6 introduced a class syntax for defining constructor functions and prototypes.

```
class Person {
    constructor(name) {
        this.name = name;
    }
    sayHello() {
        console.log(`Hello, my name is
${this.name}`);
    }
}
```

Modules

ES6 introduced a module system using the import and export keywords, allowing you to organize your code into reusable modules.

```
// math.js
export function add(a, b) {
    return a + b;
}

// main.js
import { add } from './math';
```

Promises

Promises provide a cleaner way to work with asynchronous operations.

```
function fetchData() {
    return new Promise((resolve, reject) =>
 {
        // Asynchronous code
        if (dataReceived) {
            resolve(data);
        } else {
            reject(error);
        }
    });
}
```

Iterators and Generators

ES6 introduced iterators and generators, which make it easier to work with collections and asynchronous code.

```
// Iterator example
for (const item of myArray) {
    console.log(item);
}

// Generator example
function* generatorFunction() {
    yield 1;
    yield 2;
    yield 3;
}
```

MERN

Stack Developer

Let’s connect

– Next Part Coming Soon . . .

Joy Chandra Das

@joychandradas

Share More Knowledge Let’s Connect