ES6 Javascript Cheatsheet You Will *Regret Missing*

Mohd Saalim
@md-saalim

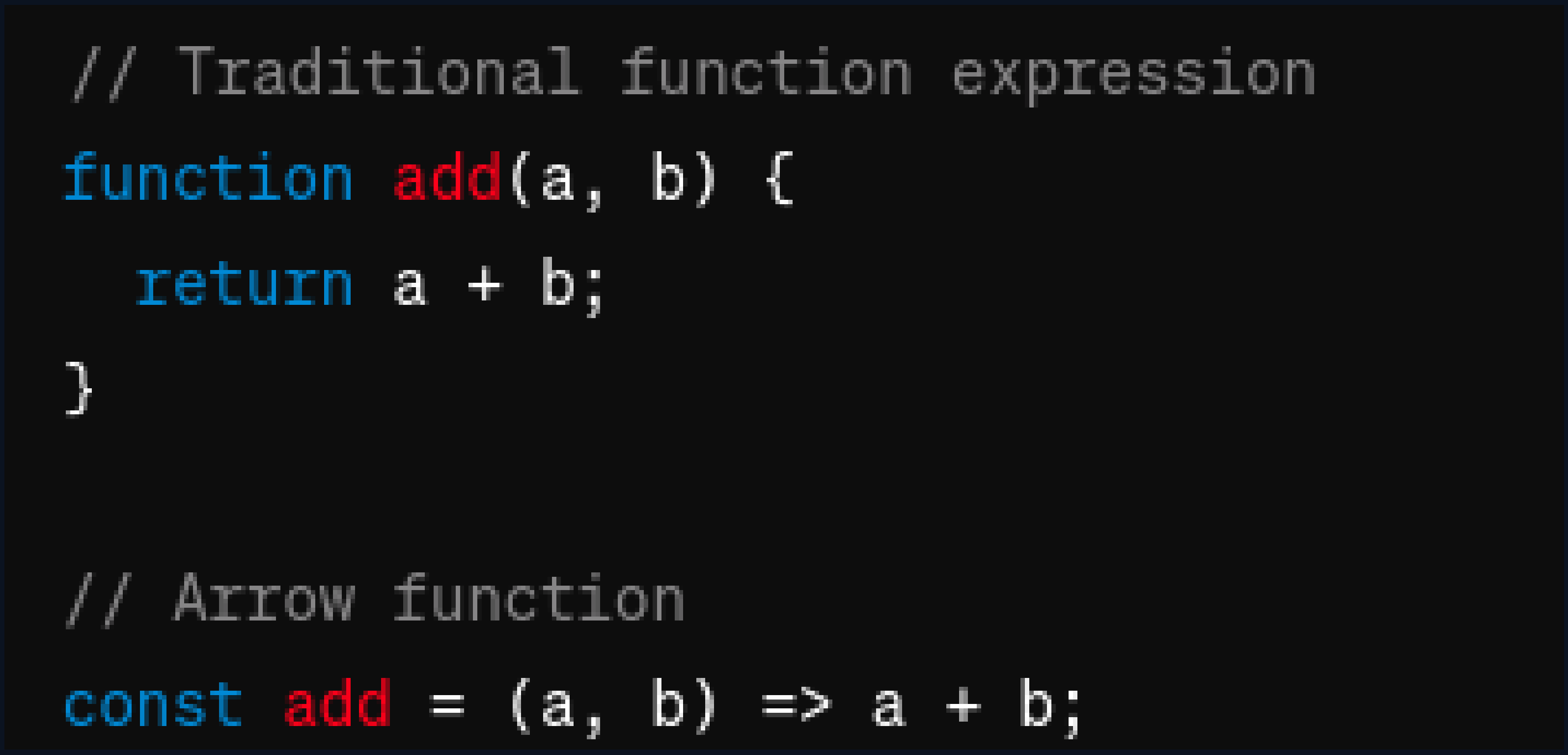
Introduction to ES6

- ES6, or ECMAScript 2015, is a major update to the JavaScript language. It introduces new features and enhancements that make JavaScript more expressive and powerful.
- Before ES6, JavaScript lacked modern features like arrow functions, classes, destructuring, and async/await, making it less expressive and more difficult to write and maintain code efficiently.

Mohd Saalim
@md-saalim

Arrow Functions

Arrow functions are a more concise syntax for writing functions in JavaScript. They provide a shorter syntax than traditional function expressions and automatically bind the this keyword. Ex:

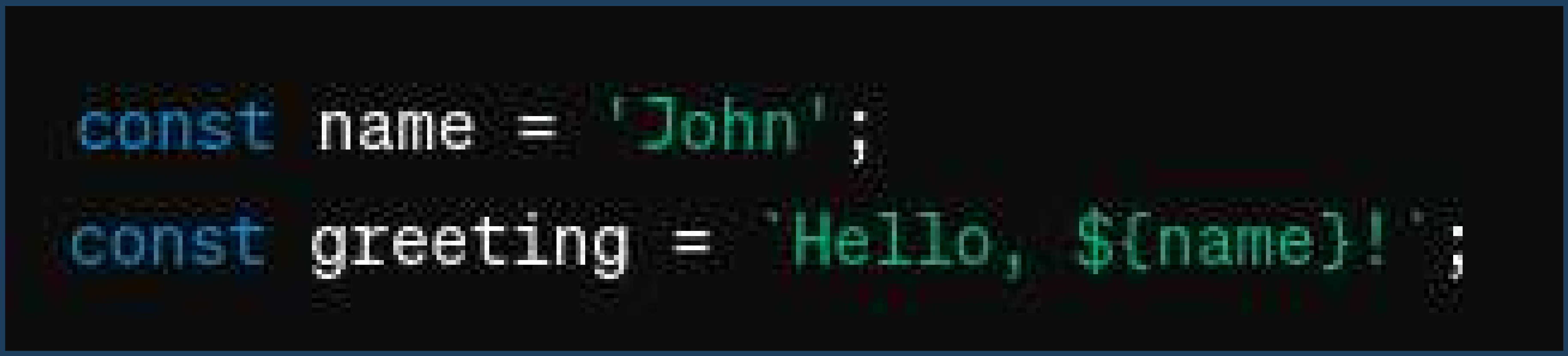


```
// Traditional function expression
function add(a, b) {
  return a + b;
}

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

Mohd Saalim
@md-saalim

Template Literals

Template literals allow you to embed expressions inside strings using backticks (`). This makes it easier to create dynamic strings. Ex:



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

Mohd Saalim
@md-saalim

Destructuring

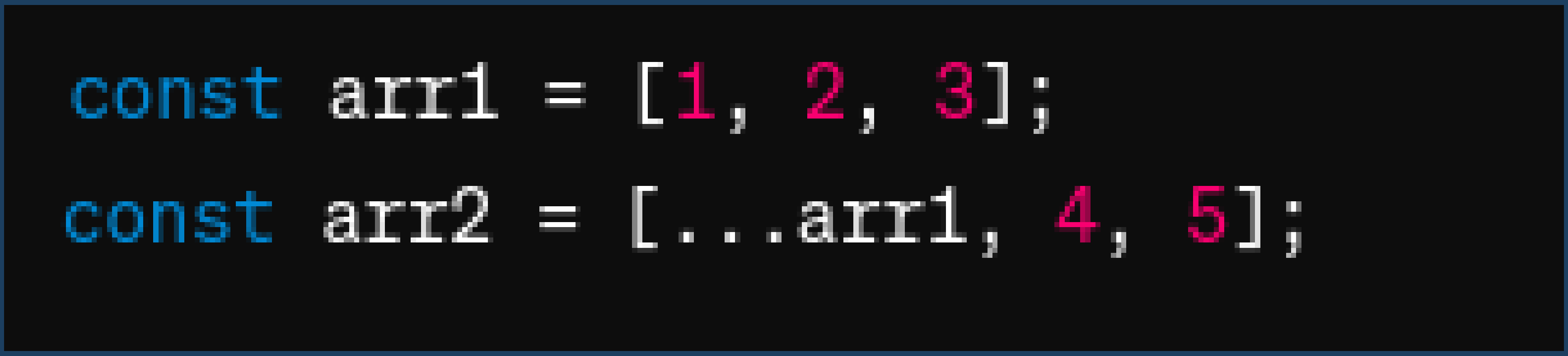
Destructuring allows you to extract values from arrays or objects into individual variables. It's a more concise and readable way to access nested data structures. Ex:

```
const person = { name: 'John', age: 30 };
const { name, age } = person;
```

Mohd Saalim
@md-saalim

Spread Operator

The spread operator (...) allows you to spread elements of an array or properties of an object into another array or object. It's useful for copying, merging, and modifying arrays and objects. Ex:



```
const arr1 = [1, 2, 3];
const arr2 = [...arr1, 4, 5];
```

Mohd Saalim
@md-saalim

Rest Parameters

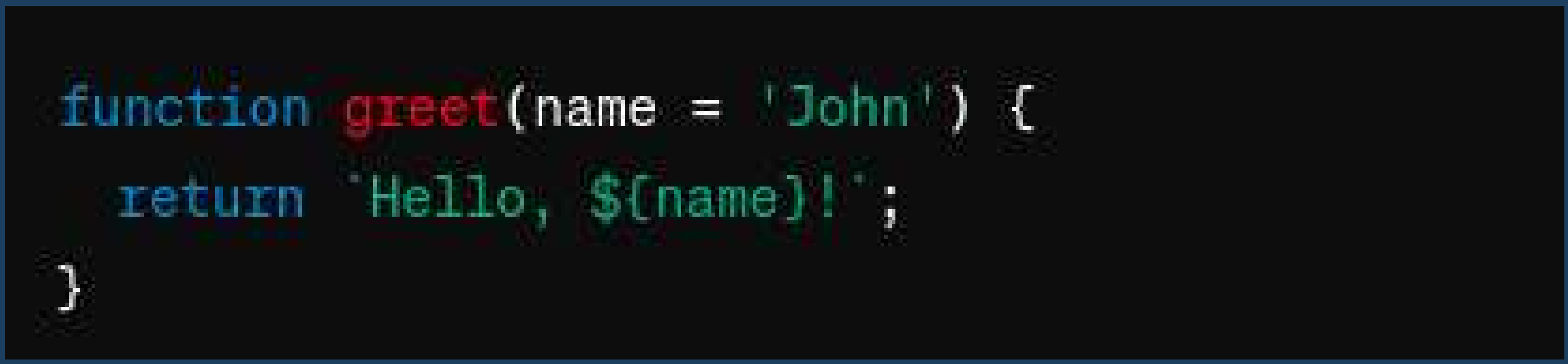
Rest parameters allow you to represent an indefinite number of arguments as an array. They're useful for functions that accept a variable number of arguments. Ex:

```
function sum(...numbers) {
  return numbers.reduce((total, num) => total + num, 0);
}
```

Mohd Saalim
@md-saalim

Default Parameters

Default parameters allow you to provide default values for function parameters. If a value is not passed, the default value will be used. Ex:

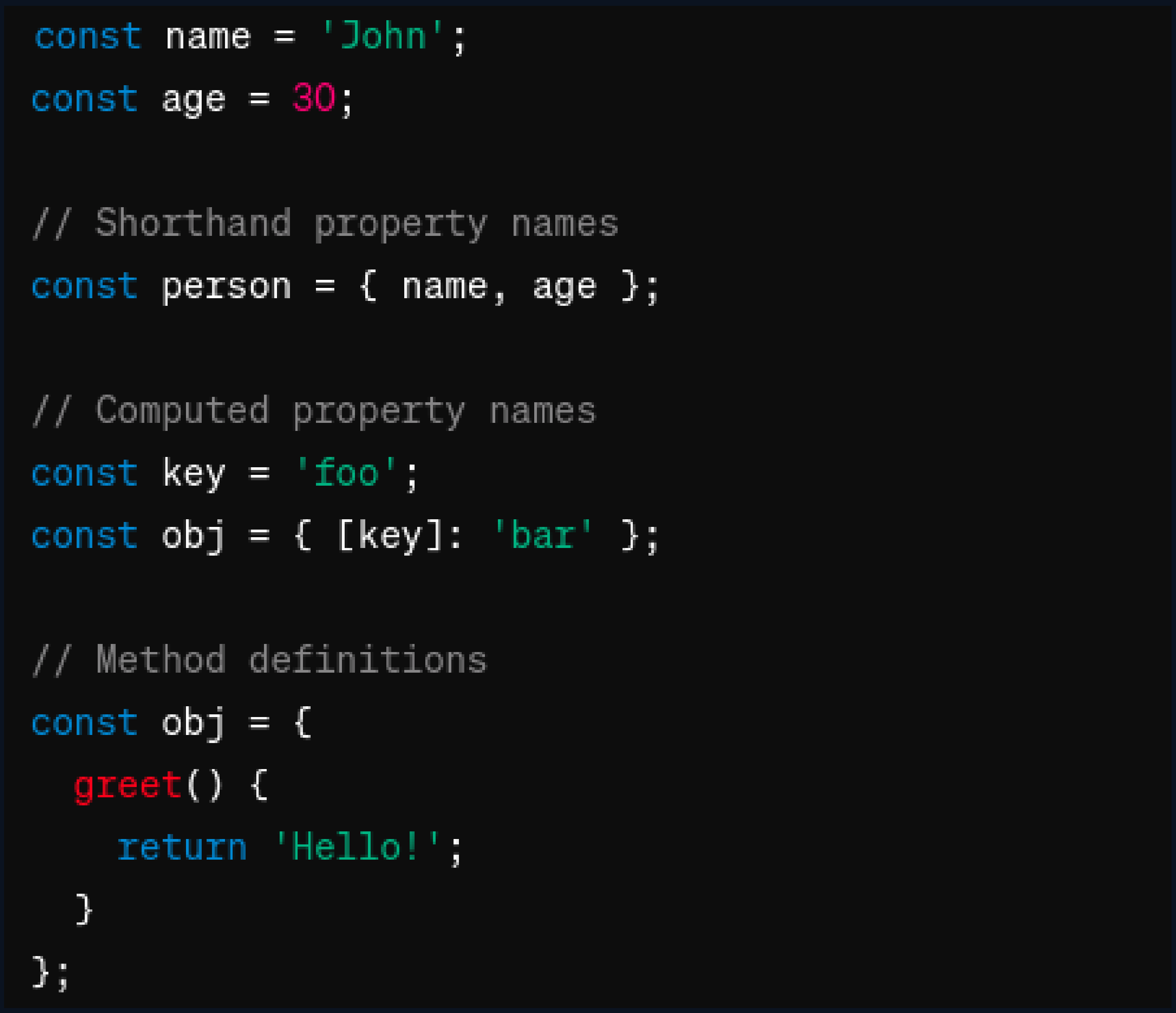


```javascript
function greet(name = 'John') {
  return `Hello, ${name}!`;
}
```

Mohd Saalim
@md-saalim

Object Literal Enhancements

Object literal enhancements provide new ways to create and work with objects. This includes shorthand property names, computed property names, and method definitions. Ex:

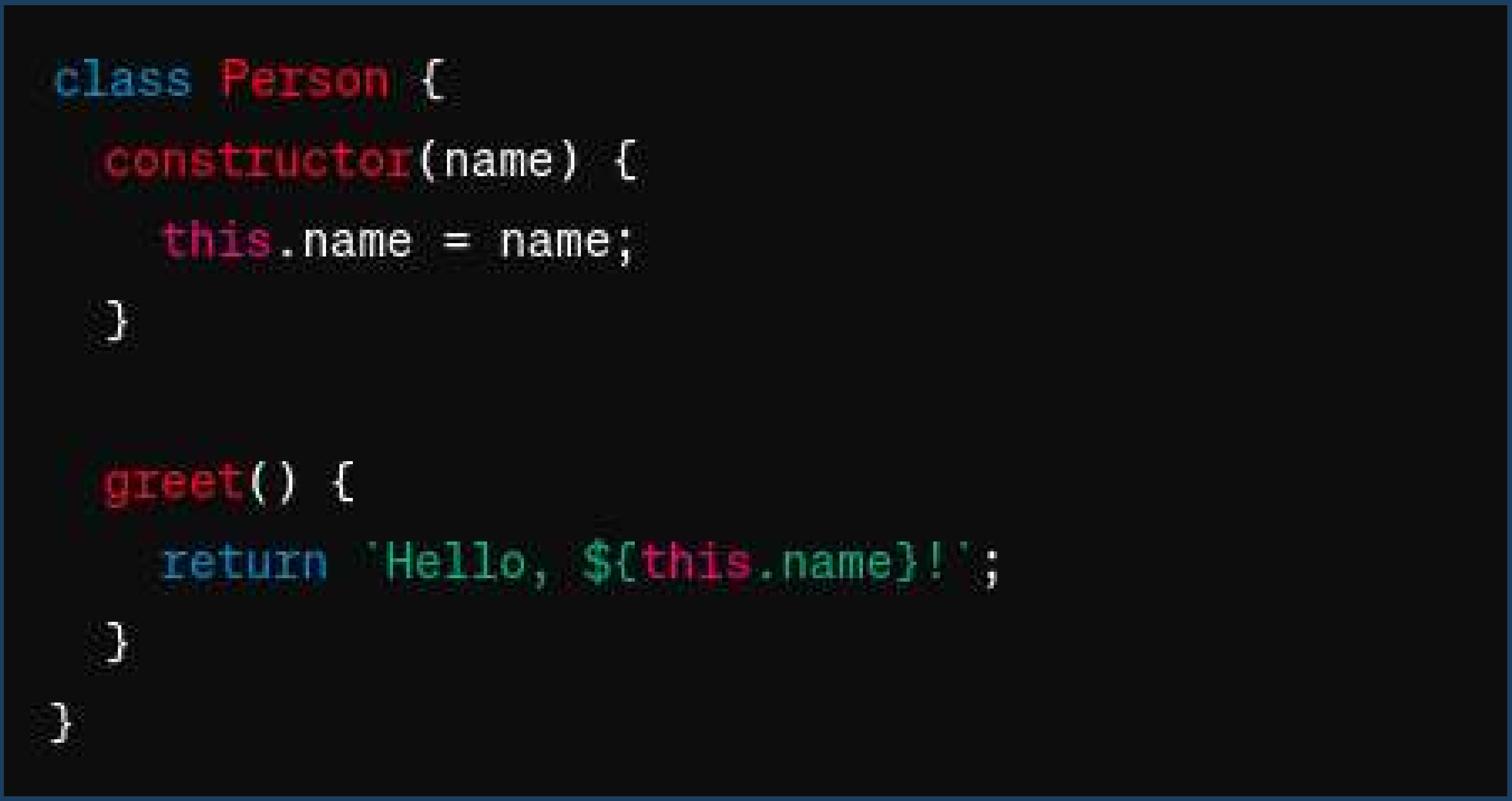


```javascript
const name = 'John';
const age = 30;

// Shorthand property names
const person = { name, age };

// Computed property names
const key = 'foo';
const obj = { [key]: 'bar' };

// Method definitions
const obj = {
  greet() {
    return 'Hello!';
  }
};
```

Mohd Saalim
@md-saalim

Classes

Classes provide a more natural and object-oriented way to create and work with objects in JavaScript. They're syntactic sugar over the existing prototype-based inheritance model. Ex:



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

  greet() {
    return `Hello, ${this.name}!`;
  }
}
```

Mohd Saalim
@md-saalim

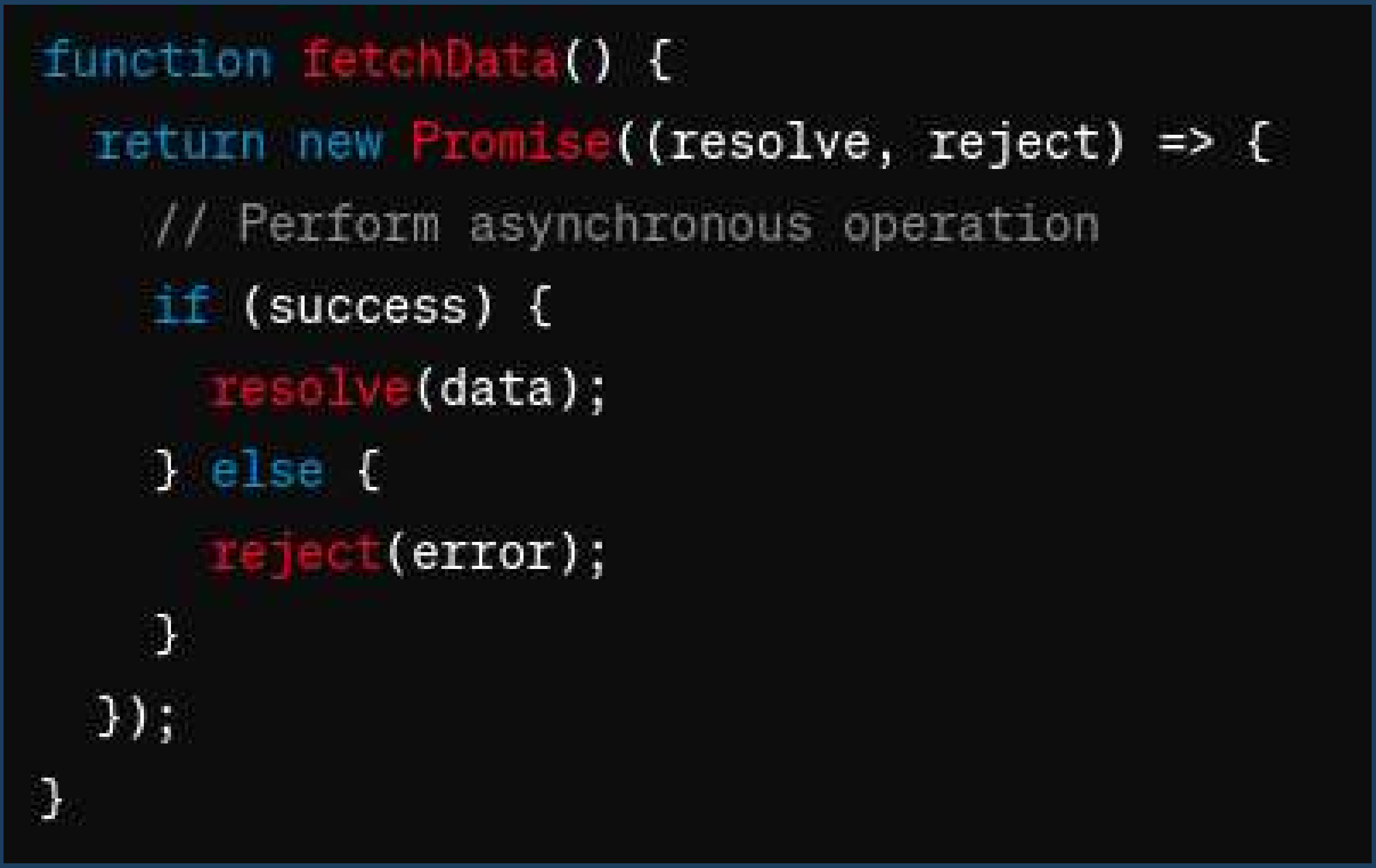
Modules

ES6 introduces a new way to organize and structure code using modules. Modules allow you to import and export functions, classes, and variables between different files. Ex:

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

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

Promises

Promises provide a better way to handle asynchronous operations in JavaScript. They represent a value that might be available now, or in the future, or never. Ex:



```
function fetchData() {
  return new Promise((resolve, reject) => {
    // Perform asynchronous operation
    if (success) {
      resolve(data);
    } else {
      reject(error);
    }
  });
}
```

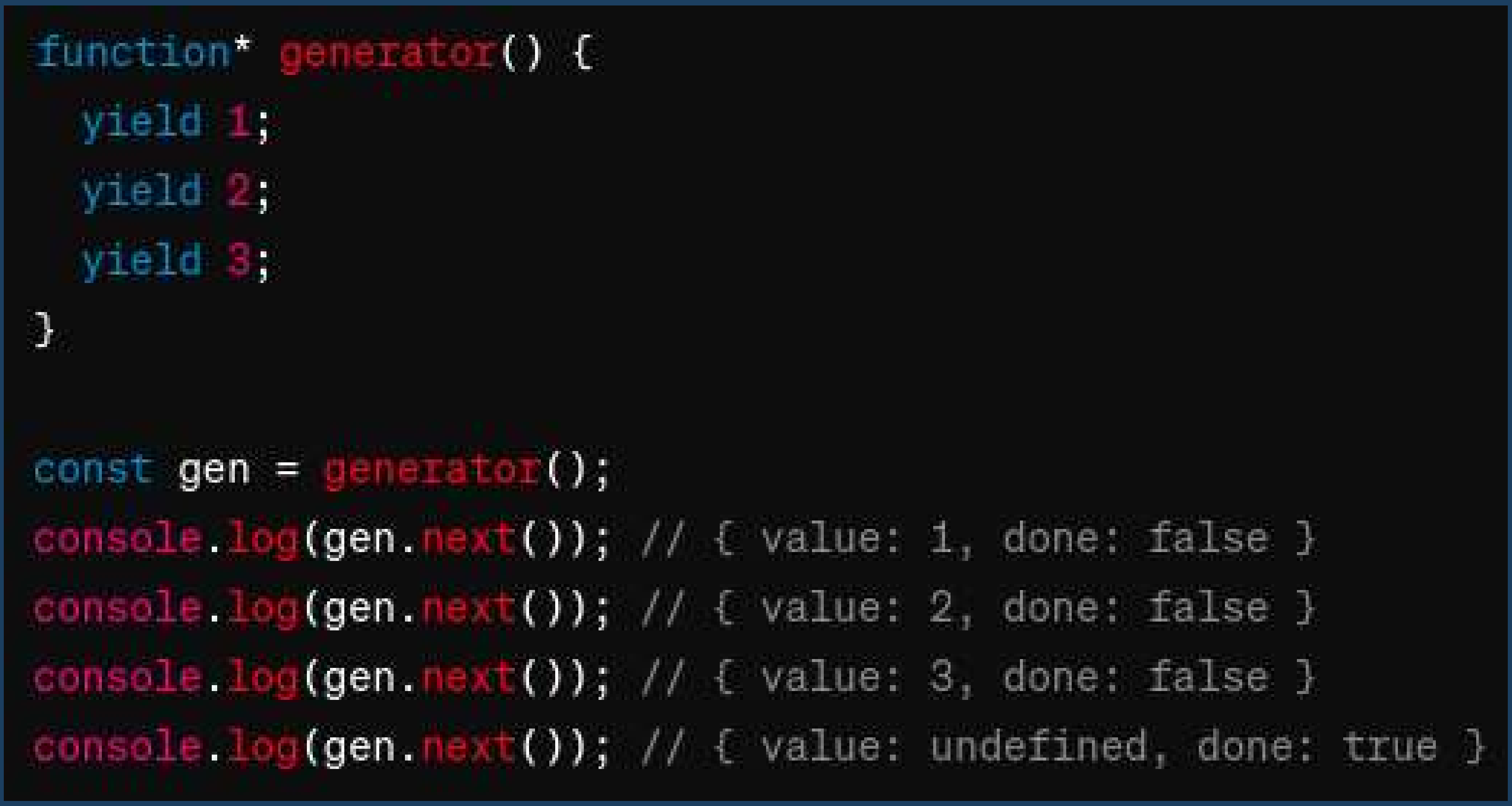
Async/Await

Async/await is a syntactic sugar over promises that makes working with asynchronous code easier and more readable. It allows you to write asynchronous code that looks synchronous. Ex:

```javascript
async function fetchData() {
  try {
    const data = await fetch('https://api.example.com/data');
    console.log(data);
  } catch (error) {
    console.error(error);
  }
}
```

Generators

Generators are a new type of function in ES6 that allow you to pause and resume the execution of a function. They're useful for creating iterators and asynchronous operations. Ex:



```
function* generator() {
  yield 1;
  yield 2;
  yield 3;
}

const gen = generator();
console.log(gen.next()); // { value: 1, done: false }
console.log(gen.next()); // { value: 2, done: false }
console.log(gen.next()); // { value: 3, done: false }
console.log(gen.next()); // { value: undefined, done: true }
```

Map, Set, WeakMap, WeakSet

ES6 introduces new data structures: Map, Set, WeakMap, and WeakSet. They provide better ways to store and manage collections of data. Ex:

```
// Map
const map = new Map();
map.set('name', 'John');
console.log(map.get('name')); // John

// Set
const set = new Set([1, 2, 3]);
console.log(set.has(2)); // true

// WeakMap
const weakMap = new WeakMap();
weakMap.set(object, 'data');

// WeakSet
const weakSet = new WeakSet();
weakSet.add(object);
```

Do you have any suggestion or query?

“Comment Below”

Mohd Saalim

@md-saalim