ECMASCRIPT MODERN FEATURES

TechRacine Academy

https://techracine.com

https://academy.techracine.com

Frontend Developer Intermediate Path

Table of Contents

1. Introduction to ECMAScript

ECMAScript is the official standard on which JavaScript is based.

Modern JavaScript features are introduced through ECMAScript updates.

ES6, released in 2015, was one of the biggest updates to JavaScript.

Modern frontend frameworks such as React, Angular, and Vue heavily depend on ECMAScript features.

Understanding modern JavaScript syntax improves code readability, maintainability, and performance.

2. ES6 and Modern JavaScript

ES6 introduced major improvements to JavaScript.

Modern JavaScript development heavily depends on ES6+ features.

Developers use modern syntax to write cleaner and more maintainable code.

```
console.log("Modern JavaScript");
```

3. let and const

let and const were introduced to replace var in many scenarios.

let supports block scope.

const prevents reassignment and improves code safety.

```
let age = 25;

const company = "TechRacine";
```

4. Template Literals

Template literals simplify string concatenation.

They support multiline strings and embedded expressions.

```
const name = "John";

console.log(`Welcome ${name}`);
```

5. Arrow Functions

Arrow functions provide shorter function syntax.

They are heavily used in modern frameworks and callbacks.

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

6. Default Parameters

Functions can now define default parameter values.

This improves flexibility and reduces manual checks.

```
function greet(name = "Guest") {
    console.log(name);
}
```

7. Destructuring

Destructuring extracts values from arrays and objects.

It improves readability and simplifies assignments.

```
const employee = {

    name: "John",
    role: "Developer"

};

const { name, role } = employee;
```

8. Spread and Rest Operators

Spread operator expands arrays and objects.

Rest operator collects multiple values into arrays.

These operators improve data manipulation.

```
const nums = [1,2,3];

const newNums = [...nums, 4];
```

9. Enhanced Object Literals

Modern JavaScript simplifies object creation.

Object literals become cleaner and more readable.

```
const name = "John";

const employee = {

    name,

    greet() {
        console.log("Hello");
    }

};
```

10. Classes and Objects

ES6 introduced class syntax for object-oriented programming.

Classes improve organization and reusability.

```
class Employee {

    constructor(name) {
        this.name = name;
    }

}

const emp = new Employee("John");
```

11. Modules

Modules split code into reusable files.

Modern applications heavily depend on modular architecture.

```
export function greet() {

    console.log("Hello");

}
```

12. Promises

Promises handle asynchronous operations.

They improve readability compared to nested callbacks.

```
const promise = new Promise((resolve, reject) => {

    resolve("Success");

});
```

13. Async and Await

Async and await simplify asynchronous programming.

They improve readability and maintainability.

```
async function loadData() {

    const response = await fetch("api/data");

}
```

14. Optional Chaining

Optional chaining prevents errors when accessing undefined properties.

It improves safety and readability.

```
console.log(user?.profile?.name);
```

15. Nullish Coalescing

Nullish coalescing provides fallback values.

It works specifically for null and undefined.

```
const name = null;

console.log(name ?? "Guest");
```

16. Array Modern Methods

Modern JavaScript introduced powerful array methods.

map, filter, reduce, and find are heavily used.

```
const nums = [1,2,3];

const doubled = nums.map(n => n * 2);
```

17. Map and Set

Map stores key-value pairs.

Set stores unique values.

These collections improve data handling.

```
const users = new Set();

users.add("John");
```

18. Error Handling

Error handling improves application stability.

try-catch blocks handle runtime issues safely.

```
try {

    console.log(data);

}
catch(error) {

    console.log(error);

}
```

19. Best Practices

Use const whenever possible.

Prefer arrow functions for callbacks.

Write modular and reusable code.

Keep asynchronous code clean.

Use meaningful variable names.

20. Common Mistakes

Avoid excessive global variables.

Avoid deeply nested callbacks.

Do not misuse async-await.

Keep code readable and maintainable.

21. Real World Usage

Modern ECMAScript features are used in React, Angular, Vue, Node.js, SaaS products, AI dashboards, and enterprise applications.

Frontend engineers heavily rely on modern JavaScript syntax every day.

22. Summary

Modern ECMAScript significantly improved JavaScript development.

Understanding ES6+ features improves code quality and developer productivity.

Modern frontend engineering strongly depends on ECMAScript concepts.

23. Exercises

1. Rewrite old JavaScript using let and const.

2. Practice template literals.

3. Create arrow functions.

4. Use destructuring with objects.

5. Practice spread operators.

6. Create a class-based application.

7. Use promises with API calls.

8. Practice async-await.

9. Build modular JavaScript files.

10. Practice modern array methods.