Let's dive into **modern JavaScript (ES6+)**, which refers to the features introduced in **ECMAScript 2015 (ES6)** and later versions. These features make JavaScript more concise, readable, and powerful. I'll break it down in a structured way with examples.

---

1. Variable Declarations: `let` and `const`

- **`let`**: Block-scoped variable. Can be reassigned.
- **`const`**: Block-scoped constant. Cannot be reassigned.

```
let age = 25;
age = 26; // □ works

const name = "Alice";
name = "Bob"; // □ error
```

---

2. Arrow Functions

Shorter syntax for functions, with lexical `this`.

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

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

console.log(add(2, 3)); // 5
```

---

3. Template Literals

Multi-line strings and embedded expressions using backticks `` ` ``.

```
const name = "Alice";
const greeting = `Hello, ${name}! Welcome to ES6.`;
console.log(greeting); // Hello, Alice! Welcome to ES6.
```

---

4. Destructuring

Extract values from arrays or objects easily.

```
// Array destructuring
const numbers = [1, 2, 3];
```

```
const [a, b] = numbers;
console.log(a, b); // 1 2

// Object destructuring
const user = { name: "Alice", age: 25 };
const { name, age } = user;
console.log(name, age); // Alice 25
```

---

5. Default Parameters

Provide default values for function parameters.

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

greet(); // Hello, Guest!
greet("Alice"); // Hello, Alice!
```

---

6. Spread & Rest Operators

- **Spread (`...`)**: Expand iterable into individual elements.
- **Rest (`...`)**: Collect multiple elements into an array.

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

// Rest
function sum(...numbers) {
  return numbers.reduce((a, b) => a + b, 0);
}
console.log(sum(1, 2, 3, 4)); // 10
```

---

7. Modules

Use `export` and `import` to structure code.

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

// app.js
import { add } from './math.js';
console.log(add(2, 3)); // 5
```

---

8. Promises & Async/Await

Handle asynchronous operations more cleanly than callbacks.

```
// Promise
const fetchData = () => {
  return new Promise((resolve, reject) => {
    setTimeout(() => resolve("Data received"), 1000);
  });
};

fetchData().then(console.log);

// Async/Await
async function getData() {
  const data = await fetchData();
  console.log(data);
}
getData();
```

---

9. Classes

ES6 introduces a class syntax for object-oriented programming.

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

  greet() {
    console.log(`Hi, I'm ${this.name}`);
  }
}

const alice = new Person("Alice", 25);
alice.greet(); // Hi, I'm Alice
```

`for...of` Loop: Iterates over iterable objects like arrays.

```
const arr = [10, 20, 30];
for (const num of arr) {
  console.log(num);
}
```