Day 6 → JavaScript ES6+ and Modules:

1. Arrow Functions:
Arrow functions are a concise syntax for writing functions in JavaScript. They provide a more compact and expressive way to define functions. Here's an example:

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

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

2. Template Literals:
Template literals allow you to create dynamic strings by embedding expressions and multiline strings. They are enclosed in backticks (` `) instead of single or double quotes. Here's an example:

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

console.log(greeting); // Output: Hello, John!

const multilineString = `
```

**This is a
multiline string.
`;**

```
console.log(multilineString);
// Output:
// This is a
// multiline string.
```

3. Destructuring:
Destructuring is a convenient way to extract values from arrays and objects into individual variables. It simplifies the process of accessing nested data. Here's an example:

```javascript
// Array destructuring
const numbers = [1, 2, 3];
const [a, b, c] = numbers;

console.log(a); // Output: 1
console.log(b); // Output: 2
console.log(c); // Output: 3

// Object destructuring
const person = { name: 'John', age: 30 };
const { name, age } = person;

console.log(name); // Output: John
console.log(age); // Output: 30
```

```

**4. Modules and Imports/Exports:**
**ES6 modules allow you to organize your code into separate files and easily share functionality between them. Here's an example:**

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

// main.js module
import { add, subtract } from './math.js';

console.log(add(5, 3)); // Output: 8
console.log(subtract(5, 3)); // Output: 2
```

5. Default Parameters:
Default parameters allow you to specify default values for function parameters if no value or undefined is passed. Here's an example:

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

greet(); // Output: Hello, Guest!
```
```

```
greet('John'); // Output: Hello, John!
```

6. Spread and Rest Operators:
The spread operator allows you to expand an iterable (e.g., an array) into individual elements, while the rest operator gathers individual arguments into an array. Here's an example:

```javascript
// Spread operator
const numbers = [1, 2, 3];
const newNumbers = [...numbers, 4, 5];

console.log(newNumbers); // Output: [1, 2, 3, 4, 5]

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

console.log(sum(1, 2, 3, 4, 5)); // Output: 15
```

7. Promises and Async/Await:
Promises and async/await are used for handling asynchronous operations in a more organized and readable manner. Here's an example:

```javascript
```

```
// Promise
function fetchData() {
  return new Promise((resolve, reject) =>

 {
    setTimeout(() => {
      resolve('Data fetched successfully!');
    }, 2000);
  });
}

fetchData().then(data => {
  console.log(data); // Output: Data fetched successfully!
});

// Async/await
async function fetchDataAsync() {
  const data = await fetchData();
  console.log(data); // Output: Data fetched successfully!
}

fetchDataAsync();
```

Now, let's bring it all together to create a project using HTML, CSS, and JavaScript. Since I don't have specific requirements, I'll provide a simple example:

1. Create an HTML file (`index.html`) with the following structure:

```html
<!DOCTYPE html>
<html>
<head>
  <title>ES6+ Project</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <h1>Welcome to the ES6+ Project!</h1>

  <script src="script.js"></script>
</body>
</html>
```

2. Create a CSS file (`styles.css`) to style your project as desired.

3. Create a JavaScript file (`script.js`) to add interactivity. Here's an example:

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

const result = add(5, 3);
const output = `The sum is ${result}`;

console.log(output);
```

```

4. Create a JavaScript module (`math.js`) to export the necessary functions. Here's an example:

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

Make sure all files are placed in the same directory, and you're ready to go! When you open the `index.html` file in a web browser, it will load the CSS styles and execute the JavaScript code.
```