Internship Report – Frontend Dev

Week 5: JavaScript Advanced Topics

Name: Zainab
Father Name: Assad Qayyum
Date: 23rd July, 2025
Internship Domain: Front-end Intern
Task: JS - ES6+ Concepts: Destructuring, Spread/Rest, Template Literals

Task Overview: (Day3)

Today's task focused on learning **JavaScript ES6+ concepts,** which include modern syntax features that make JavaScript more powerful and easier to work with. The specific topics covered were: **Destructuring, Spread Operator, Rest Operator,** and **Template Literals**.

Content Covered:

- Destructuring in JavaScript
- Spread Operator (...)
- Rest Operator (...)
- Template Literals ( )

ES6+ in JavaScript:

ES6+ refers to **ECMAScript 6 and newer versions** (ES7, ES8, ES9, etc.) — these are **modern versions of JavaScript** that introduced powerful features to make coding easier, cleaner, and more efficient.

1. Destructuring in JavaScript (Unpack values from arrays or objects)

Destructuring allows you to **unpack values** from arrays or objects and assign them to variables in a single step. It makes your code cleaner and more readable.

a) Array Destructuring

It lets you extract values from an array and assign them to individual variables based on their position in the array.

Syntax:

const [var1, var2] = array;

Example:

```
const fruits = ["apple", "banana", "cherry"];
const [first, second] = fruits;

console.log(first);  // "apple"
console.log(second); // "banana"

// Skipping values
const [,, third] = fruits;   // Skips first two
console.log(third);    // "cherry"
```

b) Object Destructuring

Object destructuring allows you to extract properties from an object and assign them to variables.

Syntax:

const {key1, key2} = object;

Example:

```
const person = { name: "Zainab", age: 20 };
const { name, age } = person;
console.log(name); // "Zainab"

// Renaming variables

const { name: fullName } = person;
console.log(fullName); // "Zainab"
```

2. Spread Operator (...)

The spread operator **expands** iterable items (like arrays, strings, or objects) into individual elements. "Expand or copy everything from arrays or objects"

a) Spread with Arrays

Spread with arrays allows you to copy or merge arrays by expanding their elements into a new array.

```
const arr1 = [1, 2, 3];
const arr2 = [...arr1, 4, 5];
console.log(arr2); // [1, 2, 3, 4, 5]
```

- Copying arrays
- Merging arrays
- Expanding arrays into function arguments

b) Spread with objects:

Spread with objects lets you create a copy of an object or add new properties by expanding its key-value pairs into another object.

```
const obj1 = { a: 1, b: 2 };
const obj2 = { ...obj1, c: 3 };

console.log(obj2); // { a: 1, b: 2, c: 3 }
```

3. Rest Operator (...)

Rest syntax collects multiple elements into a single array or object. It's used in function parameters or destructuring.

a) Rest in Functions:

It allows you to gather all extra arguments passed to a function into a single array.

Example:

```
function sumAll(...numbers) {
  return numbers.reduce((sum, num) => sum + num, 0);
}
console.log(sumAll(1, 2, 3)); // 6
```

b) Rest in Destructuring:

Array:

Collects the remaining elements of an array after the first few into a new array.

```
const [first, ...rest] = [10, 20, 30, 40];
console.log(first); // 10
console.log(rest);  // [20, 30, 40]
```

Object:

Collects the remaining properties of an object into a new object after extracting specific ones.

```
const person = { name: "Zainab", age: 20, city: "Karachi" };
const { name, ...others } = person;

console.log(name);   // "Zainab"
console.log(others); // { age: 20, city: "Karachi" }
```

4. Template Literals (`)

Template literals let you write cleaner strings with variables, expressions, and even multiline support using backticks (`).

a) String Interpolation with ${}

It lets you insert variables or expressions directly into a string using ${} inside backticks.

```
const name = "Zainab";
const age = 20;

const intro = `My name is ${name} and I am ${age} years old.`;
console.log(intro);
```

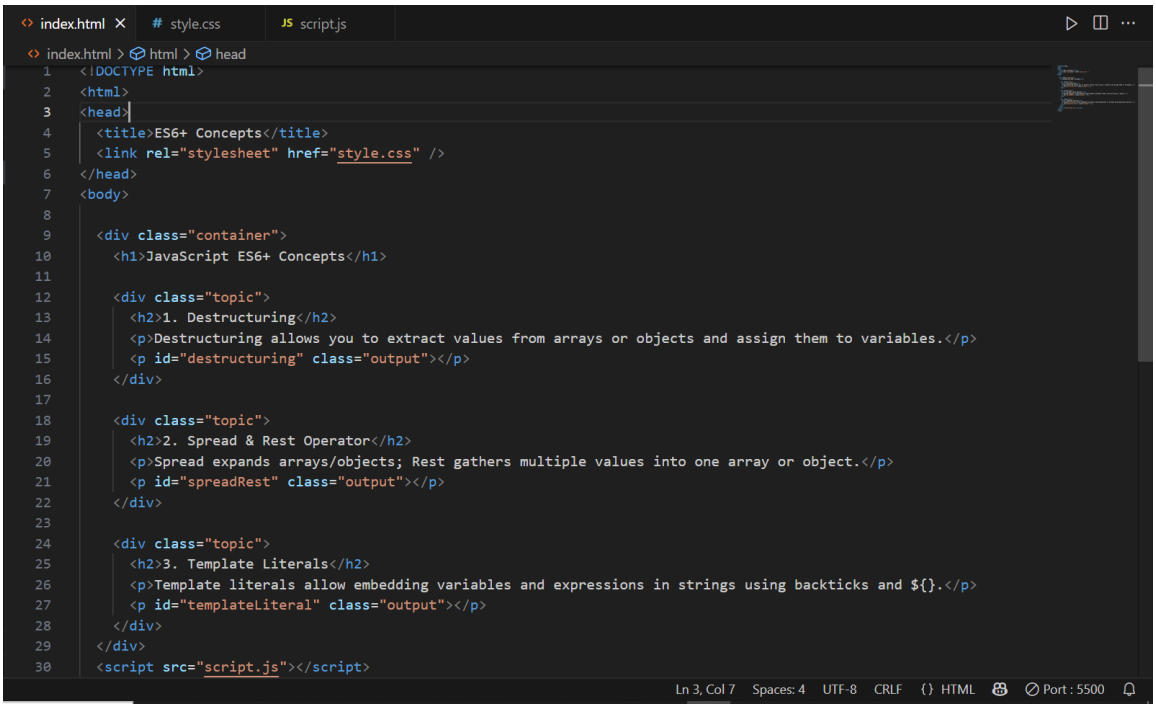
b) Multiline Strings:

It allows you to write strings across multiple lines without using newline characters or \n.

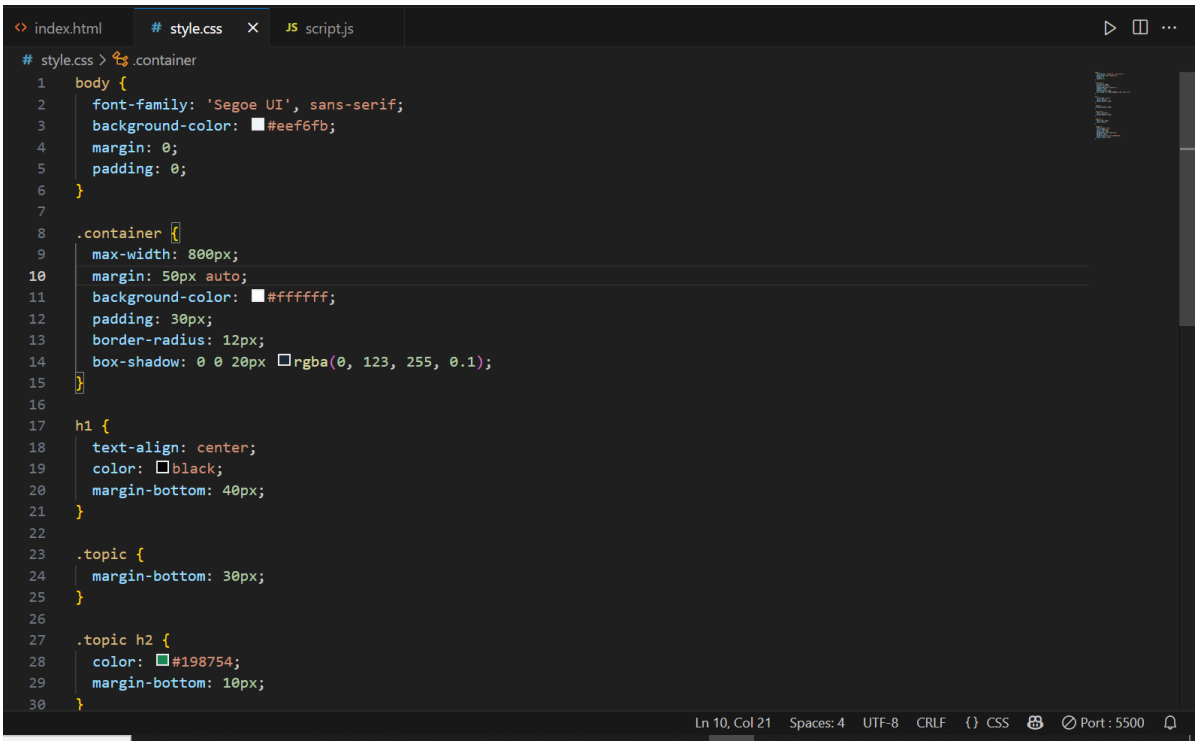
```
const message = `
Hello Zainab,

Welcome to your JavaScript advanced training!
Keep going  `;
console.log(message);
```

Practice Code:

Html:



CSS:



```css
.topic {
}

.topic h2 {
  color: #198754;
  margin-bottom: 10px;
}

.topic p {
  font-size: 16px;
  color: #444;
}

.output {
  font-weight: bold;
  color: #0d6efd;
  margin-top: 10px;
  background-color: #f1f8ff;
  padding: 10px;
  border-left: 4px solid #0d6efd;
  border-radius: 6px;
}
```

JavaScript:

```javascript
// Destructuring
const user = { name: "Zainab", age: 20, city: "Abbottabad" };
const { name, ...details } = user;

// Spread
const skills = ["HTML", "CSS"];
const updatedSkills = [...skills, "JavaScript", "ES6"];

// Rest
function listSkills(first, ...others) {
  return `Primary skill: ${first}, Others: ${others.join(", ")}`;
}

// Template Literals
const intro = `My name is ${name} and I live in ${user.city}.`;

// Display Output
document.getElementById("destructuring").innerHTML = `Destructuring used to get name: ${name}`;
document.getElementById("spreadRest").innerHTML = `Spread & Rest Example: ${listSkills(...updatedSkills)}`;
document.getElementById("templateLiteral").innerHTML = `Template Literal Intro: ${intro}`;
```

Conclusion:

Today I learned essential ES6+ JavaScript features like destructuring, spread/rest operators, and template literals. These concepts simplify code and are used heavily in modern development. The hands-on code practice helped reinforce theoretical knowledge.