1. 5 Javascript ES6 Concepts

We have already covered a lot of ES6 concepts like:

- Arrow Functions
- Javascript Modules and import/export

Now let's learn about some more ES6 concepts in detail

- Javascript Spread Operator
- Object Destructuring
- Template Literal

ES6 (ECMAScript 2015) is a major update to JavaScript that introduced powerful new features, making the language **more modern, readable, and efficient**. It significantly improved JavaScript by adding **let/const, arrow functions, template literals, classes, modules, and more**.

Brief History of ECMAScript (ES)

JavaScript follows the **ECMAScript** standard, which evolves.

Here's a quick timeline of major versions:

- **ES5 (2009)** → Added `JSON`, `strict mode`, and improved object handling.
- **ES6 (2015)** → Major update (let/const, arrow functions, classes, modules, etc.).
- **ES7 - ES2023** → Continued updates (async/await, optional chaining, etc.).

1. JavaScript Spread Operator

The spread operator (...) in JavaScript allows expanding elements of an array or object into individual elements. It's commonly used for copying, merging, and passing values.

```
function show(a, b, ...args) {
  console.log(a); // one
```

```
   console.log(b); // two
   console.log(args); // ["three", "four", "five", "six"]
}

show('one', 'two', 'three', 'four', 'five', 'six')
```

You use the **spread syntax** `...` to copy the items into a single array. For example,

```
let arr1 = ['one', 'two'];
let arr2 = [...arr1, 'three', 'four', 'five'];
console.log(arr2); // ["one", "two", "three", "four", "fiv
e"]
```

2. Object Destructuring

The destructuring assignment introduced in ES6 makes it easy to assign array values and object properties to distinct variables. For example,

```
// assigning object attributes to variables
const person = {
    name: 'Sara',
    age: 25,
    gender: 'female'
}

// destructuring assignment
let { name, age, gender } = person;

console.log(name); // Sara
console.log(age); // 25
console.log(gender); // female
```

3. Template Literal

Template literals are enclosed by backticks (`` ` ``) instead of single (`'`) or double (`"`) quotes.

- Template literals allow embedding expressions directly within strings using the `${}` syntax.
- This replaces the need for concatenation using the `+` operator.
- Template literals preserve line breaks without needing escape characters (`\n`).

```
const name = "Alice";
const age = 25;

const message = `Hello, my name is ${name}, and I am ${age} y
console.log(message);
// Output: Hello, my name is Alice, and I am 25 years old.
```

Assignments

- Extract the name and age from an object in a single line.
- Create a string using variables name and age with a template literal.