Day 7: ES7 Features - UI Demo

Arrow Functions

Overview

Arrow functions provide a concise syntax for writing functions in JavaScript. They are particularly useful for short, anonymous functions.

Example

```
// Arrow Functions
const add = (x, y) => {
  const result = x + y;
  console.log(`Result of adding ${x} and ${y}: ${result}`);
};

add(5, 7);
```

Output

Result of adding 5 and 7: 12

Task

1. Modify the **add** function to subtract two numbers.
2. Create a new arrow function that multiplies two numbers.

Template Literals

Overview

Template literals are a way to embed expressions inside string literals, providing a more flexible and readable way to create strings.

Example

```
// Template Literals
const name = 'John';
const greeting = `Hello, ${name}!`;
```

console.log(greeting);

Output

Hello, John!

Task

1. Change the value of **name** to your own name.
2. Create a new template literal that includes your name and a custom message.

Destructuring

Overview

Destructuring allows you to extract values from arrays or objects and assign them to variables in a concise way.

Example

// Destructuring

const numbers = [1, 2, 3];

const [a, b, c] = numbers;

console.log(`Destructured values: a=${a}, b=${b}, c=${c}`);

Output

Destructured values: a=1, b=2, c=3

Task

1. Create a new array with five elements and destructure it into individual variables.
2. Try destructuring an object with key-value pairs.

Let, Const, and Block Scope

Overview

ES7 introduced block-scoped variables using the **let** and **const** keywords. Unlike **var**, variables declared with **let** and **const** are scoped to the nearest enclosing block.

Example

// Block Scope

if (true) {

```
    var x = 10; // Using var (function-scoped)
    let y = 20; // Using let (block-scoped)
}

console.log(`x (using var): ${x}, y (using let): ${y}`);
```

Output

```
x (using var): 10, y (using let): 20
```

Task

1. Declare a new variable inside an **if** statement using **let** and try to access it outside the block.
2. Declare a variable with **const** and try to reassign a new value to it.