ASSIGNMENT 1

UI/UX SPECIALIST

1. **Create a function that accepts multiple numbers and returns their sum using the rest operator.**

CODE :

```
function sumNumbers(...nums) {
   return nums.reduce((a, b) => a + b, 0);
}
console.log(sumNumbers(1, 2, 3));
console.log(sumNumbers(10, 20));
console.log(sumNumbers());
```

```
Output
6
30
0

=== Code Execution Successful ===
```

2. **Write a function that accepts a string and a number. If the number is not provided, the default should be 10. The function should return the string repeated that many times.**

CODE :

```
function repeatString(str, num = 10) {
 let result = "";
   for (let i = 0; i < num; i++) {
      result += str;
   }
   return result;
}
console.log(repeatString("Hello "));
console.log(repeatString("Hi ", 3));
```

```
Output
Hello Hello Hello Hello Hello Hello Hello Hello Hello Hello
Hi Hi Hi

=== Code Execution Successful ===
```

3. **Write a JavaScript function that uses an arrow function to find the sum of all numbers in an array.**

CODE

```
const sumArray = (numbers) => {
  let sum = 0;
  for (let num of numbers) {
    sum += num;
  }
  return sum;
};
console.log(sumArray([1, 2, 3, 4, 5]));
console.log(sumArray([10, 20, 30]));
console.log(sumArray([]));
```

```
Output
15
60
0

=== Code Execution Successful ===
```

4. **Write a JavaScript function that demonstrates the differences between `let`, `const`, and `var`. Include examples of scoping issues that may occur with `var` and how `let` and `const` behave differently in block scope.**

CODE

```
function scopeDemo() {
  if (true) {
    var a = "I am var";   // Function-scoped
    let b = "I am let";   // Block-scoped
    const c = "I am const"; // Block-scoped
    console.log(b);
    console.log(c);
  }
  console.log(a); // Works because var is function-scoped
  // console.log(b); // Error(block-scoped)
  // console.log(c);  (block-scoped)
  // Reassigning values
  a = "Var can be reassigned";
  // b = "Let can be reassigned";
  // c = "Const cannot be reassigned";
}
scopeDemo();
```

```
Output
I am let
I am const
I am var

=== Code Execution Successful ===
```

Var (Function-scoped):

- **Declared variables are accessible throughout the function, even outside the block.**
- **Can be redeclared and reassigned.**
- **Scoping issue: Can cause unexpected behaviour due to lack of block scope.**

let (Block-scoped):

- **Exists only inside the block {} it was declared in.**
- **Can be reassigned but not redeclared in the same scope.**

const (Block-scoped & Immutable):

- **Also block-scoped like let.**
- **Cannot be reassigned after being declared.**