Destructuring:

- Destructuring is a JavaScript expression used to unpack values from arrays or properties from objects into distinct variables.

- Think of it like unpacking a suitcase.

```
const [a, b] = [10, 20];
const { name, age } = { name: "Manas", age: 21 };
```

Destructuring Arrays:

Basic Destructuring

```
const numbers = [1, 2, 3];
const [first, second, third] = numbers;

console.log(first); // 1
console.log(second); // 2
console.log(third); // 2
```

Destructuring With Rest Operator

```
const arr = [1, 2, 3, 4, 5];
const [a, b, ...rest] = arr; // 1 2 [3, 4, 5]
```

Default Values

```
const [x = 1, y = 2] = [10];
console.log(x, y); // 10, 2
```

Skip Array Items

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

Swapping Values

```
let a = 5, b = 10;
[a, b] = [b, a];
console.log(`a=${a}, b=${b}`); // a=10, b=5
```

Nested Arrays

```
const users = ['manas', ['muskan', 'mehek']];
const [user1, [user2, user3]] = numbers;
console.log(user1); // 'manas'
console.log(user3); // 'mehek'
```

Destructuring Objects:

Basic Destructuring

```
const obj = { name: "Manas", age: 21 };
const { name, age } = obj;
console.log(name); // "Manas"
```

Destructuring With Rest Operator

```
const obj = {
    name: "Manas",
    age: 21,
    city: 'Bhagalpur',
};
const { name, ...rest } = obj;
console.log(name); // "Manas"
console.log(rest); // { age: 21, city: 'Bhagalpur' }
```

Default Values

```
const obj = { name: "Manas", age: 21 };
const { name, city = "Unknown" } = obj;
console.log(city); // "Unknown"
```

Renaming Variables

```
const obj = { name: "Manas Kumar Lal", age: 21 };
const { name: fullName, age: years } = person;
console.log(fullName); // "Manas Kumar Lal"
```

Nested Objects

```
const user = {
    id: 1,
    profile: {
        firstName: "Manas",
        location: "India"
    }
};

const {
    id,
    profile: { firstName, location }
} = user;

console.log(id); // 1
console.log(firstName); // "Manas"
```

Destructuring in Function Parameters:

Arrays in Parameters

```
function sum([a, b]) {
    return a + b;
}

console.log(sum([5, 10])); // 15
```

Objects in Parameters

```
function greet({ name, age }) {
    console.log(`Hello ${name}, you are ${age} years old.`);
}

let obj = {
    name: "Manas",
    age: 21,
}

greet(obj);
```

1. What will be the output?

```
const arr = [1, 2, 3];
const obj = { ...arr };
console.log(obj);
```

2. How does using spread help avoid mutation? Modify the object without affecting the original?

3. Write a function that take numbers as argument and separates even and odd numbers and return an object with evens and odds and destructure the output while calling function.

4. Create a custom JavaScript function that behaves like React's useState.

   The function should:

   - Store a value (like state).
   - Return two things: the current value and a function to update it.

   Use array destructuring to extract both the value and the setter when calling your function.