Gaurav Tikekar

Front End

ES6 Destructuring

Simplify Your JavaScript Code

Swipe to right

ES6

Destructuring Arrays

- Easily extract elements from an array using array destructuring.
- The order matters, and you can skip elements if needed.
- Example: Destructuring an array to extract values:

```
const numbers = [1, 2, 3, 4];
const [first, second] = numbers;
console.log(first); // Output: 1
console.log(second); // Output: 2
```

Default Values in Array Destructuring

- Provide default values to handle cases where the array might be empty or have undefined values.
- Example: Setting default value in array destructuring:

```
const numbers = [1];
const [first = 0, second = 0] = numbers;
console.log(first); // Output: 1
console.log(second); // Output: 0 (default value)
```

Destructuring Objects

- Extract values from objects using object destructuring, based on property names.
- You can use different variable names while destructuring.
- Example: Destructuring an object to extract values:

```
const person = { name: "Alice", age: 30, city: "New York" };
const { name, age } = person;
console.log(name); // Output: Alice
console.log(age); // Output: 30
```

Default Values in Object Destructuring

- Similar to array destructuring, set default values for object properties to handle undefined or missing values.
- Example: Using default values in object destructuring:

```
const person = { name: "Bob", city: "San Francisco" };
const { name, age = 25 } = person;
console.log(name); // Output: Bob
console.log(age); // Output: 25 (default value)
```

Nested Destructuring

- Destructure nested arrays and objects for more complex data structures.
- Easily access deeply nested values with concise syntax.
- Example: Nested destructuring for a complex data structure:

```
const user = {
  name: "John",
  age: 30,
  address: {
    city: "London",
    country: "UK",
  },
};
const {
  name,
  address: { city },
} = user;
console.log(name); // Output: John
console.log(city); // Output: London
```

Rest Operator in Destructuring

- Use the rest operator (...) to collect remaining elements into a new array or object.
- Helpful when you want to destructure part of the data and group the rest.
- Example: Rest operator in array and object destructuring:

```
const numbers = [1, 2, 3, 4, 5];
const [first, second, ...rest] = numbers;
console.log(first); // Output: 1
console.log(second); // Output: 2
console.log(rest); // Output: [3, 4, 5]
```

Swapping Variables with Destructuring

- Easily swap the values of variables without using a temporary variable.
- A handy trick to streamline code and make it more readable.
- Example: Swapping variables using destructuring:

```
let a = 5;
let b = 10;
[a, b] = [b, a];
console.log(a); // Output: 10
console.log(b); // Output: 5
```

Function Parameter Destructuring

- Destructure function parameters for cleaner function definitions.
- Extract specific properties directly from the function's input object.
- Example: Function parameter destructuring:

```
function printUser({ name, age }) {
  console.log(`Name: ${name}, Age: ${age}`);
}
const user = { name: "Jane", age: 25 };
printUser(user); // Output: Name: Jane, Age: 25
```

Thank you!

Join me on a journey to explore the world of **Front End**, **DSA**, and **UI/UX**!

Gaurav Tikekar

Front End Engineer | UI/UX Designer

Stay tuned for twice-weekly posts filled with exciting content, tutorials, and industry updates.