JavaScript Destructuring | JavaScript Tutorial In Hindi #60

In this tutorial, we will explore the **destructuring** that is used to simplify the JavaScript programs. Everyone knows that JavaScript is a programming language that is used across different platforms ranging from web browsers, mobile devices, web servers, etc. Although this language has not changed a lot through the years when compared to other programming languages, but there are a few recent changes that we should know, some of the new features introduced in ES6, one of them is "*destructuring*" which is the main focus of this tutorial.

What is Destructuring?

Destructuring is breaking down a complex structure into simpler parts. In JavaScript, the complex structure is usually an array or an object. With the destructuring syntax, we can extract smaller fragments from arrays and objects. The destructuring syntax is also used for variable declaration or variable assignment.

Object Destructuring:-

The **object destructuring** is a useful feature of javascript to extract properties from objects and bind them to variables. An object destructuring is also capable of extracting multiple properties in one statement and can access properties from nested objects. It sets a default value if the property does not exist. Let us take a look at what problem JavaScript destructuring solves. Sometimes, we need top-level variables like:

```
const person = {
first: 'John',
last: 'Addison',
country: 'UK',
twitter: '@john_adison'};
const first = person.first;
const last = person.last;
```

We get this repetitive code over and over again, where we need to make a variable from something that is inside of an object or inside of an array. Here instead of creating multiple variables, we destructure them in a single line like:

```
const { first, last } = person;
```

This is the new destructuring syntax. The above code says: give us a variable **first** and **last**, and take it from the person object.

```
console.log(first); // John
console.log(last); // Addison
```

Similarly, if we also wanted the country, we would add the country into our code, like:

```
const { first, last, country} = person;
```

Array Destructuring:-

In array destructuring, we use an array literal on the left of an assignment expression. Each variable name on the array literal maps to the corresponding item at the same index on the destructured array. Here is a simple example:

```
var arr = ["Hello", "World"]
// destructuring assignment
var [first, second] = arr;
console.log(first); // Hello
console.log(second); // World
```

Output:-

Hello

World

In the above example, the left-hand side of the destructuring assignment is for defining what values are required to unpack from sourced variable. By using the **rest operator (...)** in array destructuring, we can put all the remaining elements of an array in a new array. Here is an example.

```
var colors = ["Violet", "Indigo", "Blue", "Green"];
// destructuring assignment
var [a,b,...c] = colors;
console.log(a);
console.log(b);
console.log(ac);
```

Output:-

Violet

Indigo

['Blue', 'Green']

Summary:-

The destructuring is a powerful feature that lets us extract properties from an object/array and bind these values to variables. The unique thing about destructuring is its concise syntax and ability to extract multiple variables in one statement.

Code file tut60.js as described in the video

```
console.log('This is tutorial 60');
// Destructuring in JavaScript

let a, b;
[a, b] = [34, 564];
// console.log(a, b);

[a, b, c, ...d] = [1,2,3,4,5,6,7,8,9,10, 11, 12, 13];
// console.log(a)
// console.log(b)
// console.log(c)
// console.log(d)


// Array Destructuring
({a, b, c, ...d} = {a: 34, b:345, c:67, d:45, e:34})
// console.log(a, b, c, d)

const fruits = ['Apple', 'Bananas', 'Mangoes'];
[a, b, c] = fruits;
// console.log(a, b, c)


// Object Destructuring
const laptop ={
    model: "HP Pavilion",
    age: "23 days",
```

```
    gender: "Male",
    net: 1233,
    start: function (){console.log('started');}
    }

const {model, age, gender, net, start} = laptop;
console.log(model, age, gender, net, start);
start()
```

Previous Next

CodeWithHarry