Destructing in JavaScript

When building software with JavaScript, arrays and objects are the first options that come to mind when declaring related or groups of values in your code because they are arguably the most flexible data types in JavaScript. However, although the objects or arrays often contain related values, you rarely need the whole object or array at once; you only use a part of the object or arrays at a time. Digging deep into the array or object to get the necessary value or properties can be very frustrating.

Benefits of destructuring

- readable code
- performance
- handling of default values
- handling of nested properties
- Better handling of function parameters

Array destructuring

The basic syntax for array destructuring uses square brackets on the left-hand side of an assignment statement and the array on the right-hand side.

```
let numbers = [1, 2, 3, 4, 5];
let [firstNumber] = numbers;
console.log(firstNumber); // returns 1
```

Multiple elements

extract multiple elements from an array and assign them to multiple variables

```
let numbers = [1, 2, 3, 4, 5];
let [firstNumber, secondNumber, thirdNumber] = numbers;
console.log(firstNumber, secondNumber, thirdNumber); // returns 1 2 3
```

Nested arrays

extract elements from nested arrays

```
let nestedArrs = [[1,2], [3,4], [5,6]];
let [firstArr, secondArr, thirdArr] = nestedArrs;
console.log(firstArr, secondArr, thirdArr);
```

Object destructuring

```
// Property to variable
// const {prop} = object

let car = { brand: "Honda", color: "red"};
let { brand } = car;
console.log(brand); // returns "Honda"
```

Multiple properties

You can also extract multiple properties from an object and assign them to multiple variables

```
// Multiple properties
// const { prop1,prop2, ..., propN } = object

const car = {
    brand: 'Ford',
    year: 2015,
    color: 'blue',
}
let { brand, year } = car;
console.log( brand, year ); // returns "Ford" and "2015"
```

Nested objects

use destructuring to extract properties from nested objects

```
// let { prop1, prop2, anotherObject: { prop1 } } = person;

let address = { street: "12 Benson Street", city: "Maryland", state: "Lagos" };
let person = { name: "Adams Ade", age: 25, address };
let { name, age, address: { city } } = person;
console.log(name, age, city);
```

Default values

you can destructure the object and add default values for properties that might not be present in the object.

```
// let { prop = 'Default'} = object

const job = {
  role: "Software Engineer",
  salary: 200000,
  applicationLink: "meta.com/careers/SWE-role/apply"
};
let { role, salary, isRemote = false } = job;
console.log(isRemote);
```

In addition to default values, you can also change the name of a property using an alias

```
// let { prop : myProp } = object

const job = {
  role: "Software Engineer",
  salary: 200000,
  applicationLink: "meta.com/careers/SWE-role/apply"
};
let { role: jobTitle, salary, isRemote = false } = job;
console.log(jobTitle);
```

Rest items

It is used to gather the remaining items in an array or an object into a new array or object.
You can print the first three elements and remaining elements that are in the array

```
const top20Books = [
    "1. To Kill a Mockingbird",
    "2. The Great Gatsby",
    "3. The Catcher in the Rye",
    "4. The Lord of the Rings",
    "5. The Hobbit",
    "6. The Diary of a Young Girl",
    "7. The Grapes of Wrath",
    "8. Animal Farm",
    "9. 1984",
    "10. The Adventures of Huckleberry Finn",
    "11. The Adventures of Tom Sawyer",
    "12. The Iliad",
    "13. The Odyssey",
    "14. The Republic",
    "15. The Inferno",
    "16. The Divine Comedy",
    "17. The Canterbury Tales",
    "18. Pride and Prejudice",
    "19. Jane Eyre",
    "20. Wuthering Heights"
  ];

  let [firstBook, secondBook, thirdBook, ...remainingBooks] = top20Books

  console.log(`Top three books on the list are ${firstBook}, ${secondBook}, and
${thirdBook}`

  console.log(...remainingBooks)
```

Mixed destructuring

Mixed destructuring is a combination of both object destructuring and array destructuring; it allows you to extract values from arrays and objects in a single destructuring assignment.

```
const user = {
  name: "William Benson",
  age: 20,
  address: {
    city: "Maryland",
    state: "Lagos"
  },
  hobbies: ["Swimming", "Golf", "Writing"]
};

const { name, age, address: { city, state }, hobbies: [firstHobby] } = user;
console.log(`This user's name is ${name} and he is ${age} years old. He lives at
${city}, in ${state} state and one of his hobbies is ${firstHobby}`)
```

Destructuring function parameters

JavaScript allows you to extract values from an object or array passed as a parameter to a function.

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

greet({ name: "Peter", age: 50 }); // returns Hello, Peter. You are 50 years old.
```

```
const top10Books = [
    "1. To Kill a Mockingbird",
    "2. The Great Gatsby",
    "3. The Catcher in the Rye",
    "4. The Lord of the Rings",
    "5. The Hobbit",
    "6. The Diary of a Young Girl",
    "7. The Grapes of Wrath",
    "8. Animal Farm",
    "9. 1984",
    "10. The Adventures of Huckleberry Finn"
  ];

  function rateBooks([firstBook, secondBook, thirdBook, ...remainingBooks]) {
    console.log(`Top three books on the list are ${firstBook}, ${secondBook}, and
${thirdBook}`)
    console.log(`The runner-ups are: ${[...remainingBooks]}`)
  }
  rateBooks(top10Books)
```

The above creates an array of `top10Books` and a function that logs the first three books and the remaining books in the console.

> *https://javascript.info/destructuring-assignment*
> *https://dmitripavlutin.com/javascript-object-destructuring/*
> *https://www.honeybadger.io/blog/javascript-destructuring/*