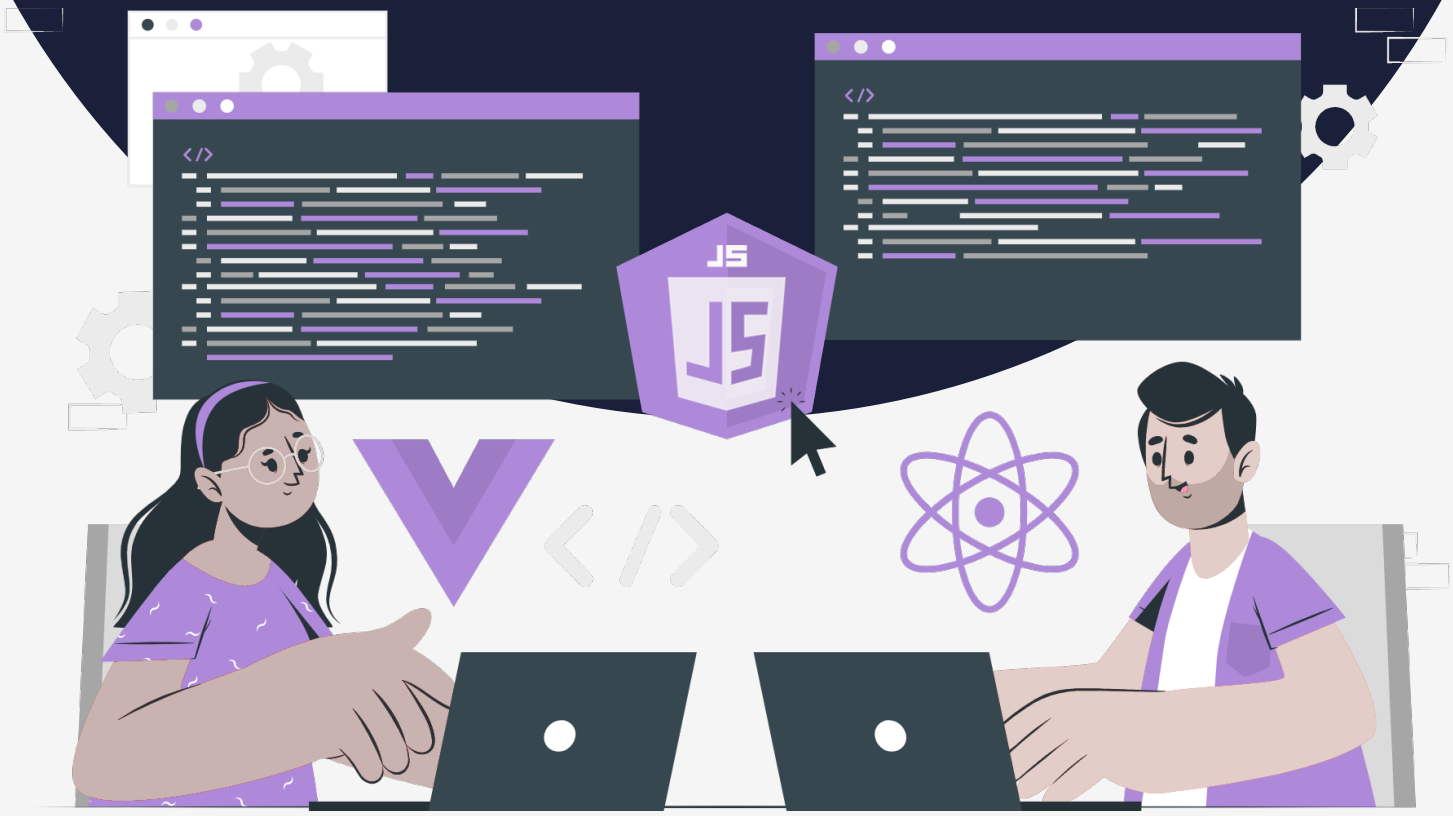
Lesson:

Destructuring objects during iteration



Topics Covered:

1. Introduction.
2. object.entries() method.
3. Implementation.

We all know that destructuring is a powerful feature that allows us to extract values from objects or arrays and assign them to variables in a concise and more readable way. It can be particularly useful when working with objects that contain nested objects. In addition to making your code more readable, destructuring can also be used during iteration to extract values from objects and use them in a loop. By combining object destructuring with iteration, you can iterate over the properties of an object and extract only the values that you need for each iteration, making your code more efficient and readable.

Object.entries()
Sometimes we need the key-value pairs of the objects so we could operate on them easily. To do this we have Object.entries() method.

Object.entries() is a built-in method that returns an array of key-value pairs for a given object. It takes an object as its argument and returns an array of arrays, where each array item contains two elements: the key and its corresponding value.

This method is useful when you need to iterate over an object's properties in a specific order or when you want to transform an object into an array for easier processing.

```
// studentDetails object

let studentDetails = {
  name: "Mithun",
  course: "Full Stack Web Development",
  email: "mithun@pw.live",
  dashboardAccessGiven: true,
};

let keyValuePairs = Object.entries(studentDetails);
console.log(keyValuePairs);

/*
OUTPUT:
[
  [ 'name', 'Mithun' ],
  [ 'course', 'Full Stack Web Development' ],
  [ 'email', 'mithun@pw.live' ],
  [ 'dashboardAccessGiven', true ]
]
*/
```

Now let's look at destructing objects during iteration.

We are given a nested object whose outer object contains the keys as the student id and the value associated with it contains the student details such as name, email, course enrolled, and the dashboard access confirmation. To print all the student id along with the student name and the course enrolled we can use destructuring objects during iteration.

```
// studentDetails object

let studentDetails = {
  student1: {
    name: "Mithun",
    course: "Full Stack Web Development",
    email: "mithun@pw.live",
    dashboardAccessGiven: true,
  },
  student2: {
    name: "Prabir",
    course: "Full Stack Web Development",
    email: "prabir@pw.live",
    dashboardAccessGiven: true,
  },
  student3: {
    name: "Alka",
    course: "Full Stack Web Development",
    email: "alka@pw.live",
    dashboardAccessGiven: false,
  },
};

for (let [key, { name, course }] of Object.entries(studentDetails)) {
  console.log(`${key} : ${name} is enrolled to ${course}`);
}

/*
OUTPUT:

student1 : Mithun is enrolled to Full Stack Web Development.
student2 : Prabir is enrolled to Full Stack Web Development.
student3 : Alka is enrolled to Full Stack Web Development.
*/
```

In the above code, the studentDetails object contains information about several students, with each student identified by a student id (student1, student2, etc.) and an object containing their name, course, email, and whether they have been given access to a dashboard.

The for...of loop uses Object.entries() to get an array of key-value pairs from the studentDetails object, and then destructures each object value to extract the name and course properties. The extracted values are used to log a string to the console that includes the student key, name, and course.

The output of this code is a list of each student's key along with their name and enrolled course. For example, the first line of the output indicates that student1 (Mithun) is enrolled in the Full Stack Web Development course.

This technique is useful when you need to iterate over objects with complex structures, and only need to access a few properties during each iteration. Destructuring can make your code more concise and expressive, improving its readability and maintainability.