Iterators in JavaScript | JavaScript Tutorial In Hindi #51

We are going to explore the concept of **"Iterators"** in this tutorial. As its name implies, iterators are a new way to *loop* over any collection in JavaScript. Today we will conceptually understand iterators along with an example. Iterators were introduced in ES6. It is one of the popular JavaScript concepts since they are widely useful and used in various places. So, let us start with the theory.

What is an iterator?

It is an object that allows us to traverse over a list or collection. Iterators' purpose is to define the sequences and implement the iterator protocol that returns an object by using a next() method that contains the **value** and **done**.

- **done:** It is a boolean value indicating whether any more elements in the sequence could be iterated upon.
- **value:** It is the current element of the sequence.

*So, we can define iterators as an **"object that knows how to access items from a collection one at a time, while keeping track of its current position within that sequence."***

Suppose we have an array, and it contains five numbers, i.e., [1,2,3,4,5]. As we know, the Iterator object has a next() method that returns the next item in the sequence. So, when we write next(), well get the element of the array. The next() method returns an object with two properties: value and done. If there are elements present in the sequence that could be iterated, then the value contains the next element and done is set to false:

{ value: "next value", done: false }

If we call the next() method after the last value has been returned, then the next() returns the result object as follows:

{done: true: value: undefined}

Here the value of the done property, which is true, indicates that there is no more value to return, and the value of the property is set to undefined.

Here is a simple example of an iterator:

```
function myIterable() {
        let counter = 0;
        return {
            next:function(){
                if (counter < 5) {
                counter++;
                return { done: false, value: counter };
                } else {
                return { done: true, value: undefined };
              }
        }
    }
}
```

Code Explanation:-

The above code executes five steps, with the counter incrementing (counter++) every run. First, we return the value 1, then the value 2, and so on till we get the last element 5then we indicate that the end of the iteration has been reached, and the value becomes equal to undefined.

Summary:-

In this tutorial, we learned about what iterator is and how we can use it. In JavaScript, an **iterator** is an object which defines a sequence and a return value upon the end of the sequence. An iterator implements the Iterator protocol by having a next() method that returns an object with two properties. As an iterator moves over the data structure and provides the elements sequentially, the object returned by the iterator contains a value and a done property.

Code as described/written in the video

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
        integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
    <title>Document</title>
</head>
```

```
<body>
    <div class="container">
        <h1 id="heading" class='yourhead rhia is'> Welcome to Code With Harry</h1>
        <div id="myfirst" class="child red good" id="first">child 1

            <ul class="this" id='myul'>
                <li class="childul" id='fui'>this</li>
                <li class="childul">is</li>
                <li class="childul">a</li>
                <li class="childul">list </li>
                <li class="childul" id='lui'>of my dreams</li>
        </div>
        <div class="child">child 2</div>
        <div class="child red">child 3</div>
        <div class="child">child 4</div>
        <form action="none.html" method="post">
            <a href="//codewithharry.com">Go to Code With Harry</a>
            <br>
            <br>
            Search this website: <input type="text" name="Hello" id="">
            <button id="btn">Submit form</button>
            <!-- <input type="button" id='btn' value="submit"> -->
        </form>
    </div>
    <!-- <br>
    <div class="no">this is a dummy div1</div>
    <div class="no">this is a dummy div2</div>
    <div class="no">this is a dummy div3</div> -->
    <div class="container">
        <h1>Student list</h1>
        <ul id="students"></ul>

        <button id="myBtn" class="btn btn-primary">Your Button</button>
        <button class="btn btn-primary">Fetch Data</button>
        <div id="content"></div>
    </div>

    <div class="container my-3">
```

```
        <h1>Pull the results by clicking the button below:</h1>

        <button class="btn btn-primary" id="meanings">Get meanings</button>
        <div class="my-2">
            <ul id="defs"></ul>
        </div>
    </div>

    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
        integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
        crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
        integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
        crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
        integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
        crossorigin="anonymous"></script>
</body>
<!-- <script src="js/tut12.js"></script> -->
<!-- <script src="js/tut14.js"></script> -->
<!-- <script src="js/tut15.js"></script> -->
<!-- <script src="js/tut16.js"></script> -->
<!-- <script src="js/tut17.js"></script> -->
<!-- <script src="js/tut18.js"></script> -->
<!-- <script src="js/tut20.js"></script> -->
<!-- <script src="js/tut21.js"></script> -->
<!-- <script src="js/tut23.js"></script> -->
<!-- <script src="js/tut24.js"></script> -->
<!-- <script src="js/tut25.js"></script> -->
<!-- <script src="js/tut27.js"></script> -->
<!-- <script src="js/tut28.js"></script> -->
<!-- <script src="js/tut30.js"></script> -->
<!-- <script src="js/tut31.js"></script> -->
<!-- <script src="js/tut32.js"></script> -->
<!-- <script src="js/tut34.js"></script> -->
<!-- <script src="js/tut37.js"></script> -->
<!-- <script src="js/tut39.js"></script> -->
<!-- <script src="js/tut39b.js"></script> -->
```

```
<!-- <script src="js/tut41.js"></script> -->
<!-- <script src="js/tut43.js"></script> -->
<!-- <script src="js/tut44.js"></script> -->
<!-- <script src="js/tut 45.js"></script> -->
<!-- <script src="js/tut46.js"></script> -->
<!-- <script src="js/tut47.js"></script> -->
<!-- <script src="js/tut48.js"></script> -->
<!-- <script src="js/tut49.js"></script> -->
<!-- <script src="js/tut51.js"></script> -->
<script src="js/tut52.js"></script>

</html>
```

Code as described/written in the video

```
console.log('The file is tutorial 51');
// Iterators

function fruitsIterator(values) {
    let nextIndex = 0;
    // we will return an object
    return {
        next: function () {
            if (nextIndex < values.length) {
                // We will return below object
                return {
                    value: values[nextIndex++],
                    done: false
                }
            }
            else {
                // We will return below object with only done
                return {
                    done: true
                }
            }
        }
    }
```

```
}


const myArray = ['Apples', 'Grapes', 'Oranges', 'Bhindi'];
console.log("My array is ", myArray)

// Using the iterator
const fruits = fruitsIterator(myArray);
console.log(fruits.next().value)
console.log(fruits.next().value)
console.log(fruits.next().value)
console.log(fruits.next().value)
console.log(fruits.next().value)
```

Previous Next

CodeWithHarry