Training Day20 Report

(5 July 2024)

JavaScript Array Methods

JavaScript Array.slice() Method

The slice() method returns a new array containing a portion of the original array, based on the start and end index provided as arguments

Syntax

```
Array.slice (startIndex , endIndex);
```

Example : Cover all **slice() method** corner cases.

JavaScript

```
// Original Array
const arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

// Case 1: Extract the first 3 elements of the array
const case1 = arr.slice(0, 3);
console.log("First 3 Array Elements: ", case1);

// Case 2: Extract the last 3 array elements
const case2 = arr.slice(-3);
console.log("Last 3 Array Elements: ", case2);

// Case 3: Extract elements from middle of array
const case3 = arr.slice(3, 7);
console.log("Case 3: Extract elements from middle: ", case3);

// Case 4: Start index is greater than end index
const case4 = arr.slice(5, 2);
console.log("Case 4: Start index is greater than end index: ", case4);

// Case 5: Negative start index
const case5 = arr.slice(-4, 9);
console.log("Case 5: Negative start index: ", case5);

// Case 6: Negative end index
const case6 = arr.slice(3, -2);
console.log("Case 6: Negative end index: ", case6);

// Case 7: Only start index is provided
const case7 = arr.slice(5);
```

```
console.log("Case 7: Only start index is provided: ", case7);

// Case 8: Start index and end index are out of range
const case8 = arr.slice(15, 20);
console.log("Case 8: Start and end index out of range: ", case8);

// Case 9: Start and end index are negative
// and out of range
const case9 = arr.slice(-15, -10);
console.log("Case 9: Start and end index are negative"
    + " and out of range: ", case9);
```

Output

```
First 3 Array Elements:  [ 1, 2, 3 ]

Last 3 Array Elements:  [ 8, 9, 10 ]

Case 3: Extract elements from middle:  [ 4, 5, 6, 7 ]

Case 4: Start index is greater than end index:  []

Case 5: Negative star...
```

Javascript Array.unshift() Method

The unshift() method is used to add elements to the front of an Array.

Syntax

```
Array.unshift(item1, item2 …)
```

Example : Adding new elements to the beginning of the array using the **unshift()** method.

JavaScript

```
// Declaring and initializing arrays
let numArr = [20, 30, 40];

// Adding element at the beginning
// of an array
numArr.unshift(10, 20);
console.log(numArr);


// Declaring and initializing arrays
let strArr = ["amit", "sumit"];

// Adding element at the beginning
// of an array
strArr.unshift("sunil", "anil");
console.log(strArr);
```

Output

```
[ 10, 20, 20, 30, 40 ]
[ 'sunil', 'anil', 'amit', 'sumit' ]
```

JavaScript Array.shift() Method

The shift() method is used to remove elements from the beginning of an array

Syntax

```
Array.shift()
```

Example : Remove an element from the beginning of an array.

JavaScript

```
// Declare and initialize array
let numArr = [20, 30, 40, 50];

// Removing elements from the
// beginning of an array
numArr.shift();

console.log(numArr);


// Declare and initialize array
let strArr = ["amit", "sumit", "anil"];

// Removing elements from the
// beginning of an array
strArr.shift();

console.log(strArr);
```

Output

```
[ 30, 40, 50 ]
[ 'sumit', 'anil' ]
```

JavaScript Array map() Method

The **map() method in JavaScript** creates a new array by applying a function to each element of the original array. It skips empty elements and does not alter the original array, making it ideal for transforming data.

One notable feature of the map() method is its handling of empty elements. It automatically skips executing the callback function for these elements,

focusing solely on processing non-empty values. This behavior ensures efficient and streamlined processing, enhancing the overall performance of array transformations.

Syntax:

```
map((element, index, array) => { /* … */ })
```

Parameters:

- **element:** It is a required parameter and it holds the value of the current element.
- **index:** It is an optional parameter and it holds the index of the current element.
- **arr:** It is an optional parameter and it holds the array.

Return Value:

It returns a new array and elements of arrays are the result of the callback function.

Example : Here, we are using the `map()` method to create a new array containing the square roots of each number in the original array.

JavaScript

```
const numbers = [1, 4, 9, 16, 25];
const squareRoots = numbers.map(num => Math.sqrt(num));

console.log(squareRoots); // Output: [1, 2, 3, 4, 5]
```

Output

```
[ 1, 2, 3, 4, 5 ]
```