1. What is map() method in Array? How to use map() method?

The `map()` method in JavaScript is a powerful higher-order function used to iterate through an array and apply a given function to each element in that array. It creates a new array by taking the result of the provided function for each element.

The basic syntax of the `map()` method is:

javascript

const newArray = originalArray.map((currentValue, index, array) => {

// return result based on currentValue

});

- `currentValue`: The current element being processed in the array.

- `index` (optional): The index of the current element being processed.

- `array` (optional): The array `map()` was called upon.

Here's an example of how to use the `map()` method:

javascript

// Example 1: Doubling each number in an array

const numbers = [1, 2, 3, 4, 5];

const doubledNumbers = numbers.map((number) => {

return number \* 2;

});

console.log(doubledNumbers); // Output: [2, 4, 6, 8, 10]

```

You can also make use of arrow function shorthand for simpler operations:

```javascript

// Example 2: Squaring each number in an array using arrow function shorthand

const numbers = [1, 2, 3, 4, 5];

const squaredNumbers = numbers.map(number => number \*\* 2);

console.log(squaredNumbers); // Output: [1, 4, 9, 16, 25]

The `map()` method is powerful and widely used for transforming data in arrays without mutating the original array. It's important to note that `map()` doesn't change the original array; it creates and returns a new array based on the provided function's logic applied to each element of the original array.

5. Find the details about following methods of Array:

find(), findIndex(), filter()

Sure, here's an overview of the mentioned Array methods: `find()`, `findIndex()`, and `filter()`.

1. \*\*`find()` Method:\*\*

- The `find()` method is used to return the first element in an array that satisfies a provided testing function. It returns the value of the first element in the array that meets the condition specified in the function. If there is no element that matches the condition, it returns `undefined`.

- Syntax:

```javascript

const foundElement = array.find((element, index, array) => {

// return true if element satisfies the condition

});

```

- Example:

```javascript

const numbers = [5, 12, 8, 130, 44];

const foundNumber = numbers.find(number => number > 10);

console.log(foundNumber); // Output: 12

```

2. \*\*`findIndex()` Method:\*\*

- The `findIndex()` method is similar to `find()`, but instead of returning the element itself, it returns the index of the first element in the array that satisfies the provided testing function. If no element matches the condition, it returns `-1`.

- Syntax:

```javascript

const foundIndex = array.findIndex((element, index, array) => {

// return true if element satisfies the condition

});

```

- Example:

```javascript

const numbers = [5, 12, 8, 130, 44];

const foundIndex = numbers.findIndex(number => number > 10);

console.log(foundIndex); // Output: 1 (index of the first element > 10)

```

3. \*\*`filter()` Method:\*\*

- The `filter()` method creates a new array with all elements that pass the test implemented by the provided function. It returns a new array containing all elements that satisfy the specified condition, without changing the original array.

- Syntax:

```javascript

const newArray = array.filter((element, index, array) => {

// return true if element satisfies the condition

});

```

- Example:

```javascript

const numbers = [5, 12, 8, 130, 44];

const filteredNumbers = numbers.filter(number => number > 10);

console.log(filteredNumbers); // Output: [12, 130, 44]

```

These methods are powerful tools for working with arrays, allowing for the retrieval or filtering of elements based on specified conditions, providing flexibility in manipulating array data.