03 DAY THREE

Array Method In Javascript

Shivam Raj
www.linkedin.com/in/shivam-raj-979b9822b/

Array Methods

- push( )
- pop( )
- shift( )
- unshift( )
- concat( )
- slice( )
- splice( )
- .indexOf( )
- lastIndexOf( )
- includes( )
- find( )
- .findIndex( )
- filter( )
- map( )
- reduce( )
- forEach( )
- some( )
- every( )
- sort( )
- reverse( )
- join( )

{.js}

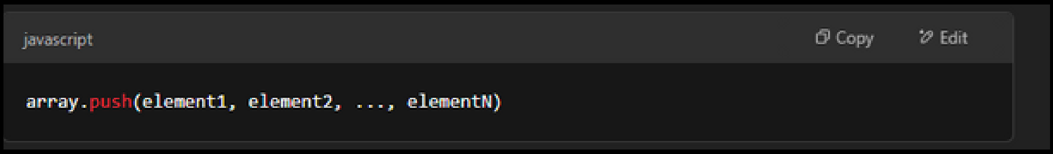
JavaScript

Shivam Raj
www.linkedin.com/in/shivam-raj-979b9822b/

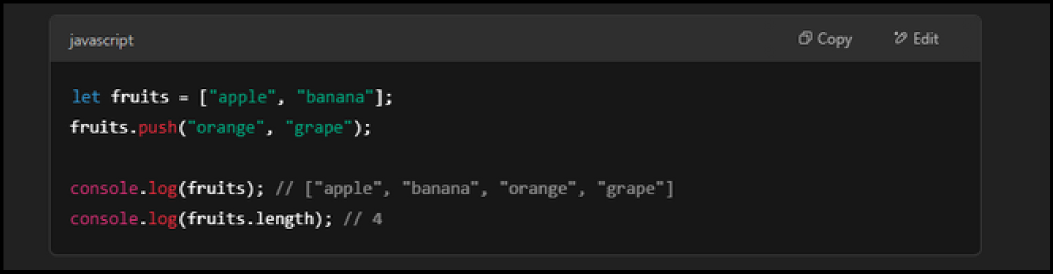
Array Methods

push( )

The push method in arrays is used to add one or more elements to the end of an array and returns the new length of the array.


javascript Copy Edit

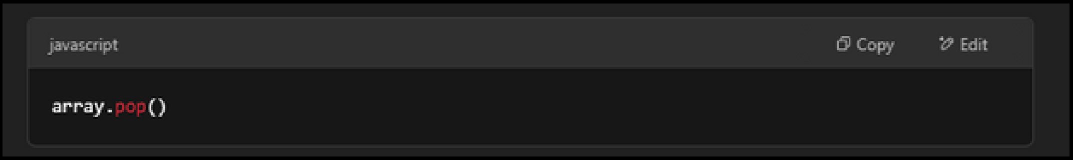
```
array.push(element1, element2, ..., elementN)
```


javascript Copy Edit

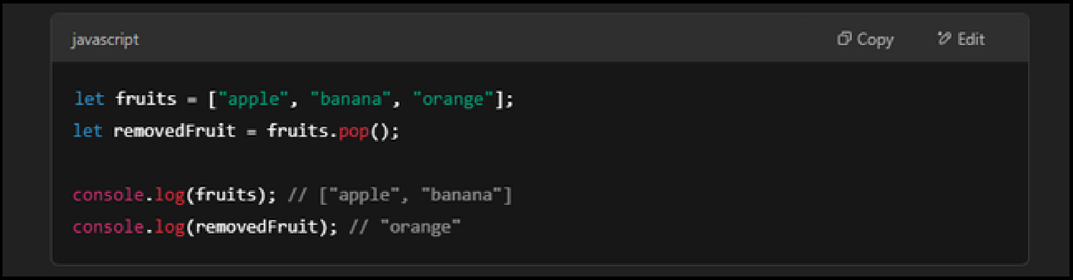
```
let fruits = ["apple", "banana"];
fruits.push("orange", "grape");

console.log(fruits); // ["apple", "banana", "orange", "grape"]
console.log(fruits.length); // 4
```

pop( )

The pop method in arrays is used to remove the last element from an array and return that removed element.


javascript Copy Edit

```
array.pop()
```

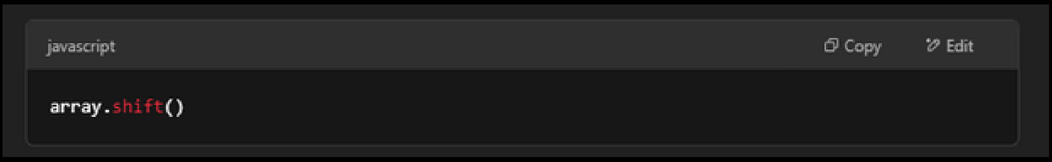

javascript Copy Edit

```
let fruits = ["apple", "banana", "orange"];
let removedFruit = fruits.pop();

console.log(fruits); // ["apple", "banana"]
console.log(removedFruit); // "orange"
```

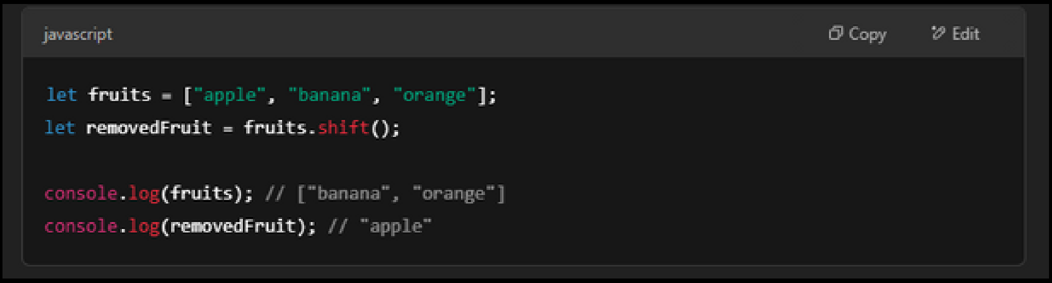
Array Methods

shift( )

The shift method in JavaScript is used to remove the first element from an array and return that removed element.



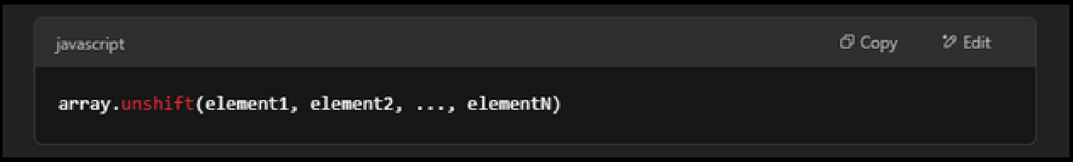
```javascript
array.shift()
```



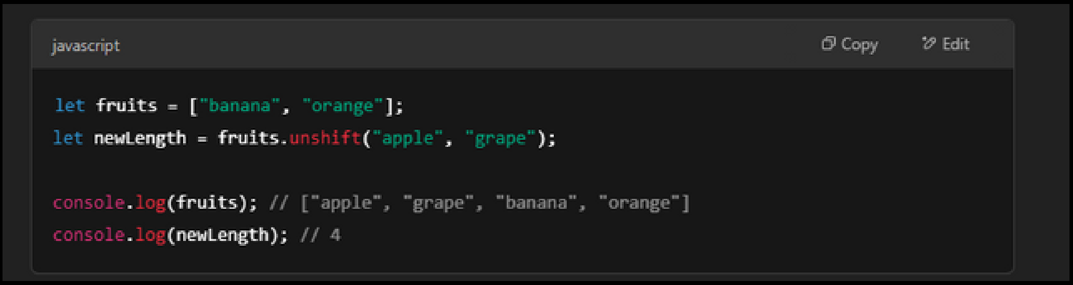
```javascript
let fruits = ["apple", "banana", "orange"];
let removedFruit = fruits.shift();

console.log(fruits); // ["banana", "orange"]
console.log(removedFruit); // "apple"
```

unshift( )

The unshift method in JavaScript is used to add one or more elements to the beginning of an array and returns the new length of the array.



```javascript
array.unshift(element1, element2, ..., elementN)
```



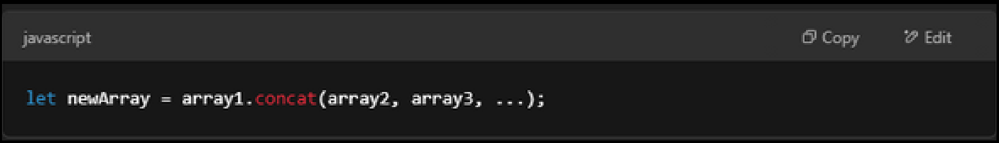
```javascript
let fruits = ["banana", "orange"];
let newLength = fruits.unshift("apple", "grape");

console.log(fruits); // ["apple", "grape", "banana", "orange"]
console.log(newLength); // 4
```

Shivam Raj
www.linkedin.com/in/shivam-raj-979b9822b/

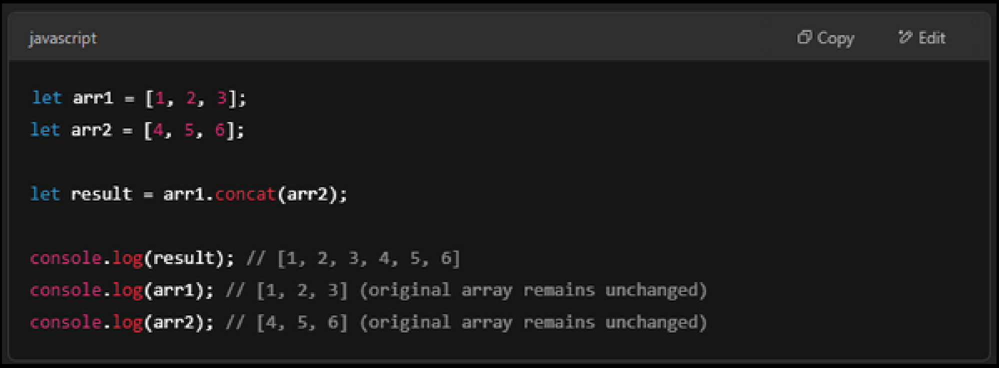
Array Methods

concat( )

The concat method in JavaScript is used to merge two or more arrays without modifying the original arrays. It returns a new array.



```javascript
let newArray = array1.concat(array2, array3, ...);
```

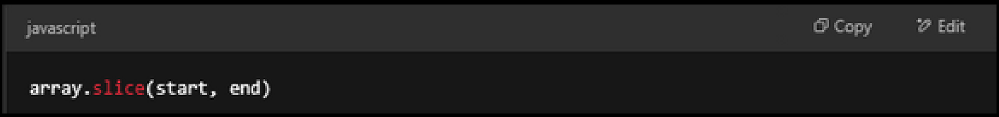


```javascript
let arr1 = [1, 2, 3];
let arr2 = [4, 5, 6];

let result = arr1.concat(arr2);

console.log(result); // [1, 2, 3, 4, 5, 6]
console.log(arr1); // [1, 2, 3] (original array remains unchanged)
console.log(arr2); // [4, 5, 6] (original array remains unchanged)
```

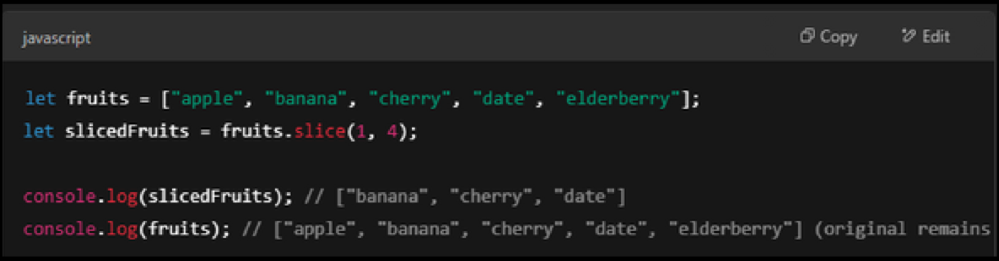
slice( )

The slice method in JavaScript is used to extract a portion of an array without modifying the original array. It returns a new array with the selected elements.

- start (optional) – The index where the extraction begins (inclusive).
- end (optional) – The index where the extraction stops (exclusive).
- (If omitted, it extracts till the end of the array.)



```javascript
array.slice(start, end)
```

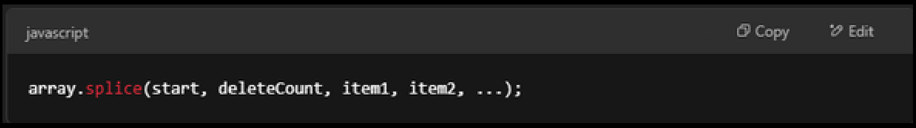


```javascript
let fruits = ["apple", "banana", "cherry", "date", "elderberry"];
let slicedFruits = fruits.slice(1, 4);

console.log(slicedFruits); // ["banana", "cherry", "date"]
console.log(fruits); // ["apple", "banana", "cherry", "date", "elderberry"] (original remains
```

Array Methods

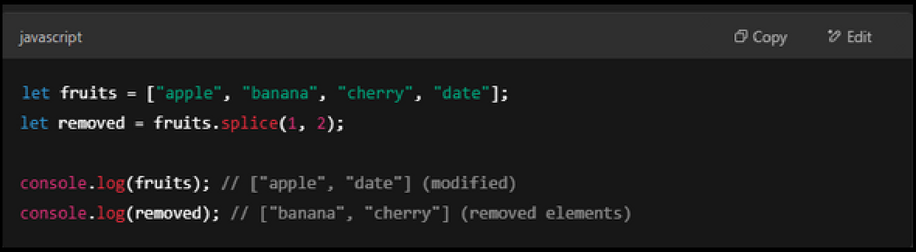
splice( )

The splice method in JavaScript is used to add, remove, or replace elements in an array. It modifies the original array and returns the removed elements.



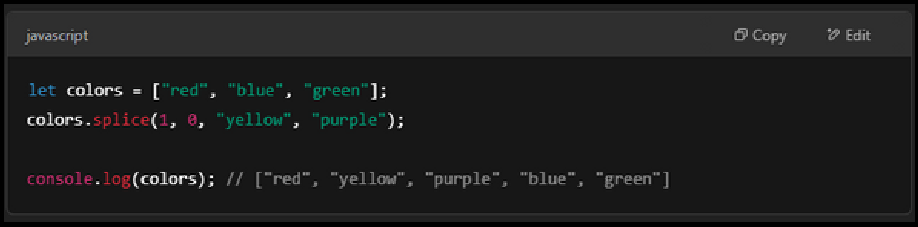
```javascript
array.splice(start, deleteCount, item1, item2, ...);
```

Remove Elements :



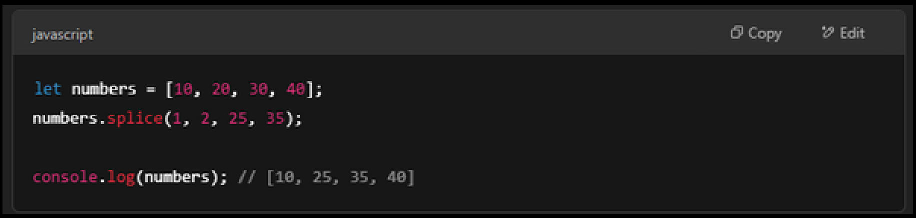
```javascript
let fruits = ["apple", "banana", "cherry", "date"];
let removed = fruits.splice(1, 2);

console.log(fruits); // ["apple", "date"] (modified)
console.log(removed); // ["banana", "cherry"] (removed elements)
```

Add Elements :



```javascript
let colors = ["red", "blue", "green"];
colors.splice(1, 0, "yellow", "purple");

console.log(colors); // ["red", "yellow", "purple", "blue", "green"]
```

Replace Element:

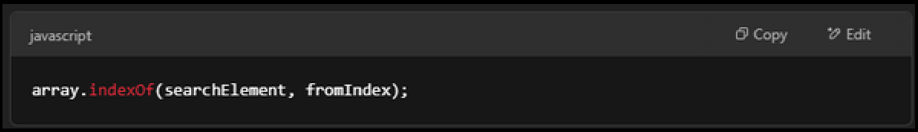


```javascript
let numbers = [10, 20, 30, 40];
numbers.splice(1, 2, 25, 35);

console.log(numbers); // [10, 25, 35, 40]
```

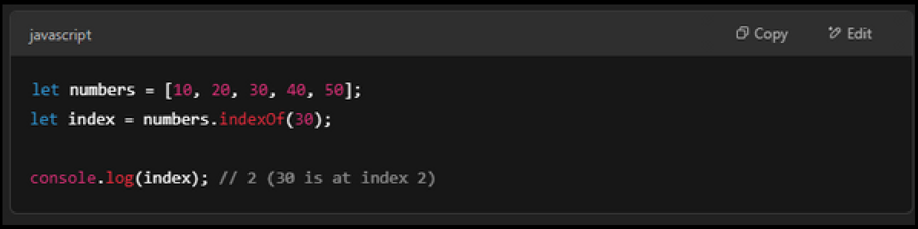
Array Methods

indexOf( )

The indexOf method in JavaScript is used to find the first occurrence of an element in an array. It returns the index of the element if found, otherwise, it returns -1.



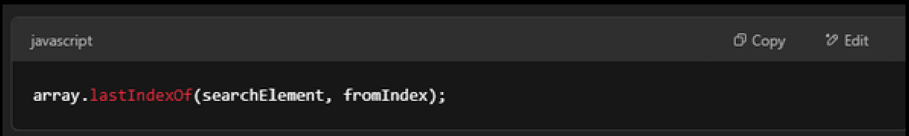
```javascript
array.indexOf(searchElement, fromIndex);
```



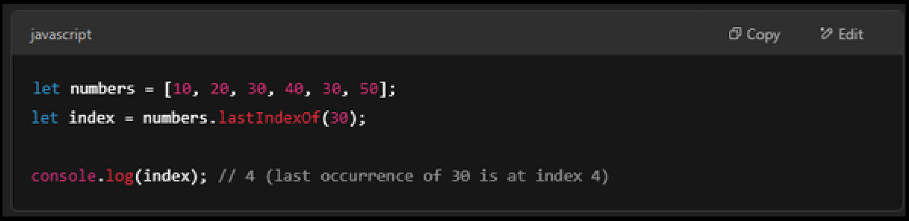
```javascript
let numbers = [10, 20, 30, 40, 50];
let index = numbers.indexOf(30);

console.log(index); // 2 (30 is at index 2)
```

lastIndexOf( )

The lastIndexOf method in JavaScript is used to find the last occurrence of an element in an array. It returns the index of the element if found; otherwise, it returns -1.



```javascript
array.lastIndexOf(searchElement, fromIndex);
```

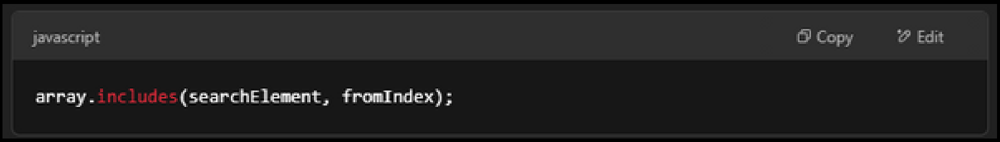


```javascript
let numbers = [10, 20, 30, 40, 30, 50];
let index = numbers.lastIndexOf(30);

console.log(index); // 4 (last occurrence of 30 is at index 4)
```

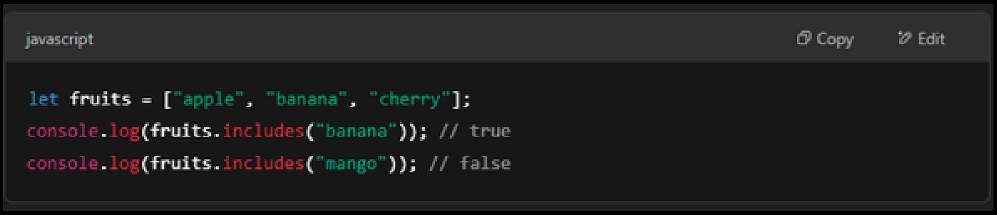
Array Methods

includes( )

The includes method in JavaScript is used to check if an array contains a specific element. It returns true if the element is found, otherwise false.



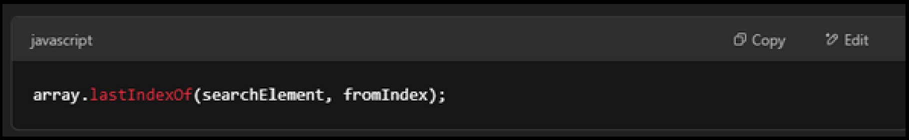
```javascript
array.includes(searchElement, fromIndex);
```



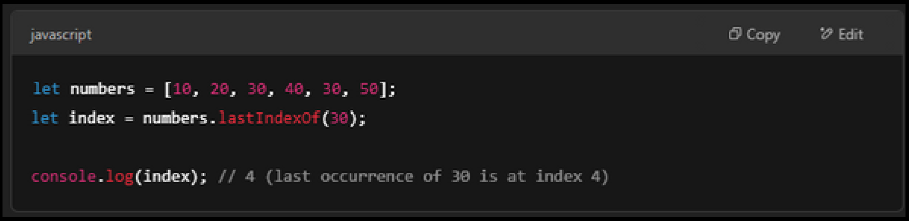
```javascript
let fruits = ["apple", "banana", "cherry"];
console.log(fruits.includes("banana")); // true
console.log(fruits.includes("mango")); // false
```

lastIndexOf( )

The lastIndexOf method in JavaScript is used to find the last occurrence of an element in an array. It returns the index of the element if found; otherwise, it returns -1.



```javascript
array.lastIndexOf(searchElement, fromIndex);
```

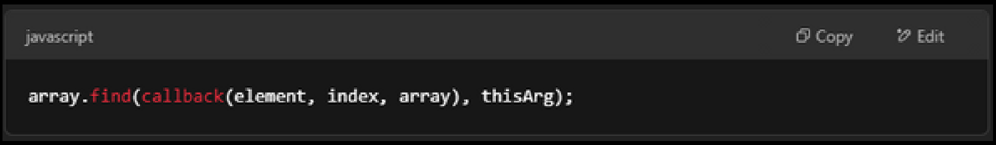


```javascript
let numbers = [10, 20, 30, 40, 30, 50];
let index = numbers.lastIndexOf(30);

console.log(index); // 4 (last occurrence of 30 is at index 4)
```

Array Methods

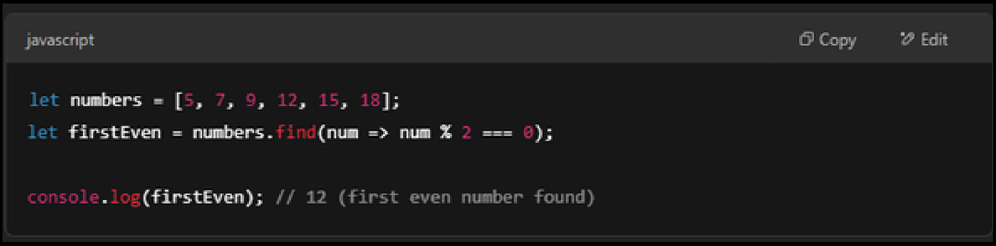
find( )

The find method is used to retrieve the first element in an array that meets a condition specified in a callback function.

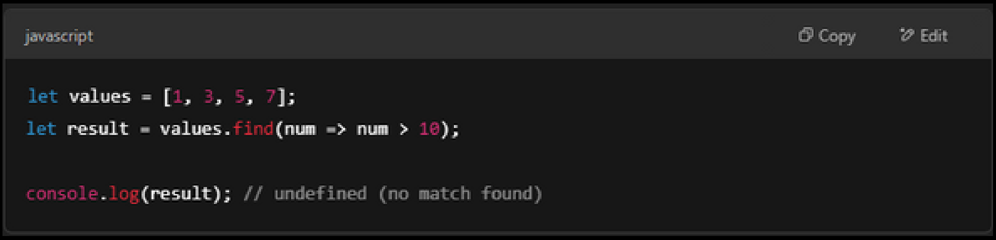


```javascript
array.find(callback(element, index, array), thisArg);
```

- callback – A function that runs on each element.
- element – The current element being processed.
- index (optional) – The index of the current element.
- array (optional) – The array find is being applied to.
- thisArg (optional) – Value to use as this in the callback.



```javascript
let numbers = [5, 7, 9, 12, 15, 18];
let firstEven = numbers.find(num => num % 2 === 0);

console.log(firstEven); // 12 (first even number found)
```

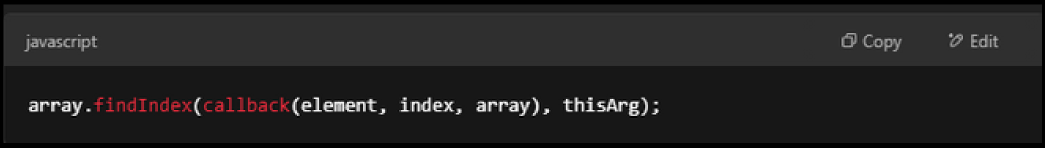


```javascript
let values = [1, 3, 5, 7];
let result = values.find(num => num > 10);

console.log(result); // undefined (no match found)
```

Array Methods

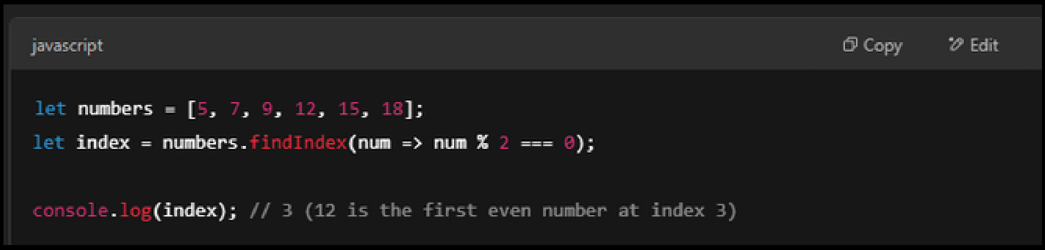
findIndex( )

The findIndex method in JavaScript is used to find the index of the first element in an array that satisfies a given condition. If no element matches, it returns -1.



```javascript
array.findIndex(callback(element, index, array), thisArg);
```

- callback – A function that tests each element.
- element – The current element being processed.
- index (optional) – The index of the current element.
- array (optional) – The array being searched.
- thisArg (optional) – Value to use as this in the callback.

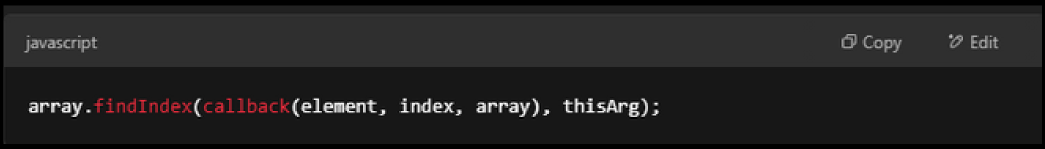


```javascript
let numbers = [5, 7, 9, 12, 15, 18];
let index = numbers.findIndex(num => num % 2 === 0);

console.log(index); // 3 (12 is the first even number at index 3)
```

Array Methods

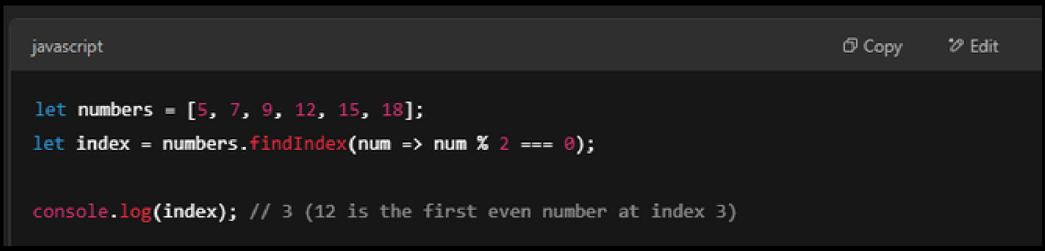
findIndex( )

The findIndex method in JavaScript is used to find the index of the first element in an array that satisfies a given condition. If no element matches, it returns -1.



```javascript
array.findIndex(callback(element, index, array), thisArg);
```

- callback – A function that tests each element.
- element – The current element being processed.
- index (optional) – The index of the current element.
- array (optional) – The array being searched.
- thisArg (optional) – Value to use as this in the callback.

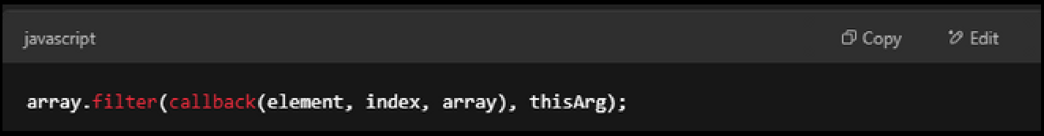


```javascript
let numbers = [5, 7, 9, 12, 15, 18];
let index = numbers.findIndex(num => num % 2 === 0);

console.log(index); // 3 (12 is the first even number at index 3)
```

Array Methods

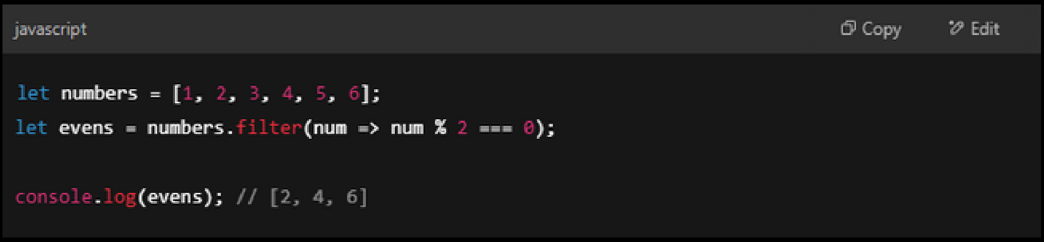
fliter()

The filter method in JavaScript is used to create a new array with elements that meet a specific condition. It does not modify the original array.

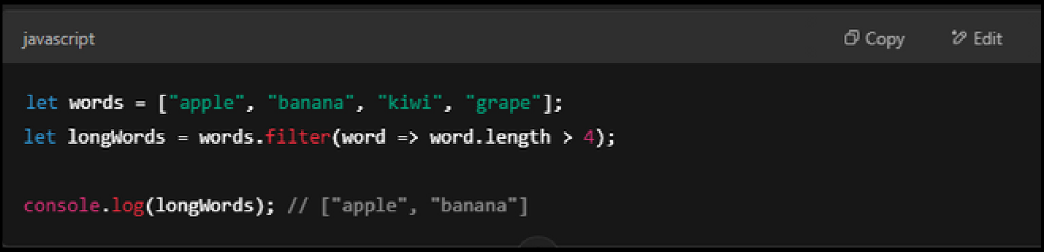


```javascript
array.filter(callback(element, index, array), thisArg);
```

- callback – A function that tests each element.
- element – The current element being processed.
- index (optional) – The index of the current element.
- array (optional) – The array being filtered.
- thisArg (optional) – Value to use as this in the callback.



```javascript
let numbers = [1, 2, 3, 4, 5, 6];
let evens = numbers.filter(num => num % 2 === 0);

console.log(evens); // [2, 4, 6]
```

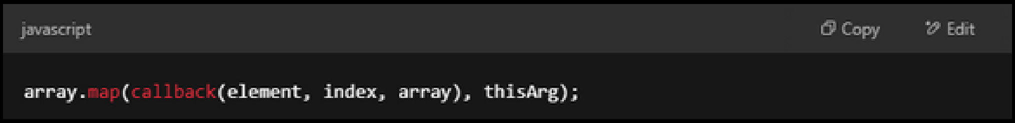


```javascript
let words = ["apple", "banana", "kiwi", "grape"];
let longWords = words.filter(word => word.length > 4);

console.log(longWords); // ["apple", "banana"]
```

Array Methods

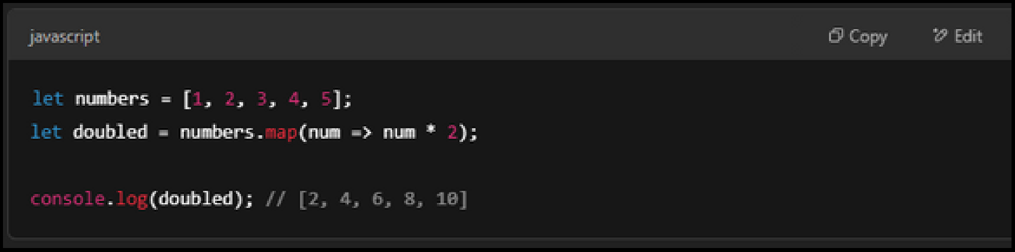
map()

The map method in JavaScript is used to create a new array by transforming each element of the original array. It does not modify the original array.



```javascript
array.map(callback(element, index, array), thisArg);
```

- callback – A function applied to each element.
- element – The current element being processed.
- index (optional) – The index of the current element.
- array (optional) – The original array.
- thisArg (optional) – Value to use as this in the callback.

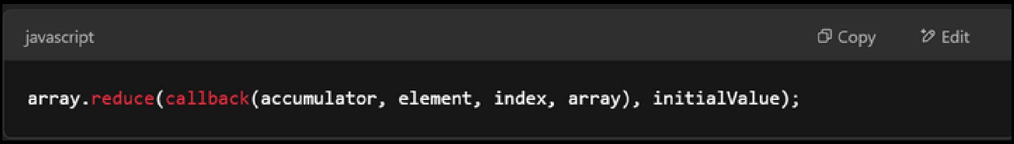


```javascript
let numbers = [1, 2, 3, 4, 5];
let doubled = numbers.map(num => num * 2);

console.log(doubled); // [2, 4, 6, 8, 10]
```

Array Methods

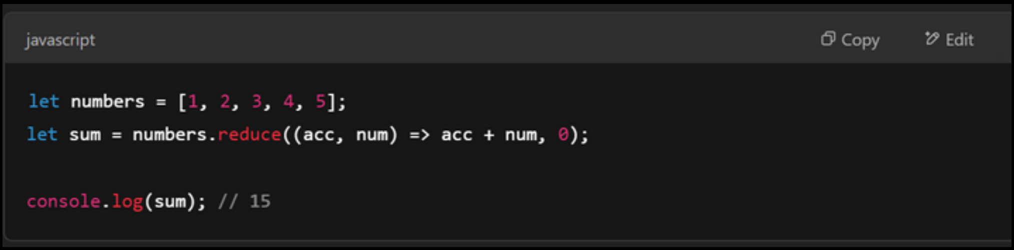
reduce( )

The reduce method in JavaScript is used to accumulate values from an array into a single result. It processes each element and carries forward an accumulated value.

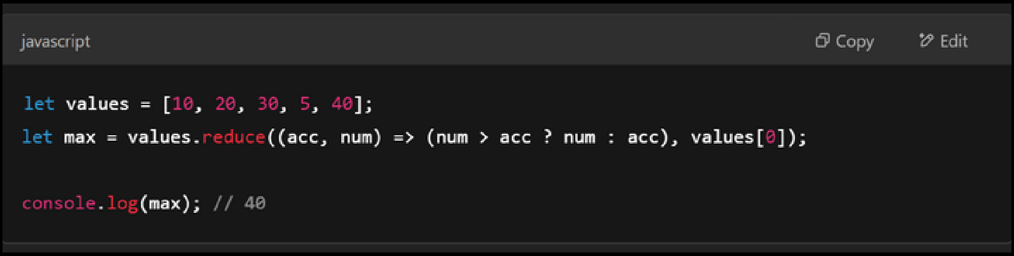


```javascript
array.reduce(callback(accumulator, element, index, array), initialValue);
```

- callback – A function executed for each element.
- accumulator – The accumulated value (previous result).
- element – The current element being processed.
- index (optional) – The index of the current element.
- array (optional) – The original array.
- initialValue (optional) – The starting value of the accumulator.



```javascript
let numbers = [1, 2, 3, 4, 5];
let sum = numbers.reduce((acc, num) => acc + num, 0);

console.log(sum); // 15
```

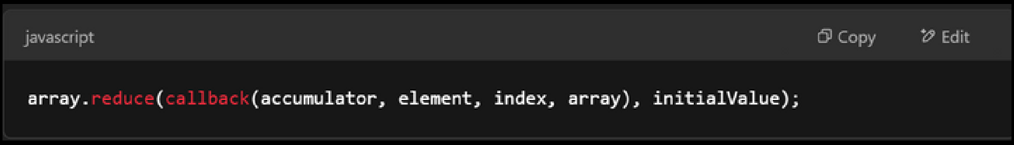


```javascript
let values = [10, 20, 30, 5, 40];
let max = values.reduce((acc, num) => (num > acc ? num : acc), values[0]);

console.log(max); // 40
```

Array Methods

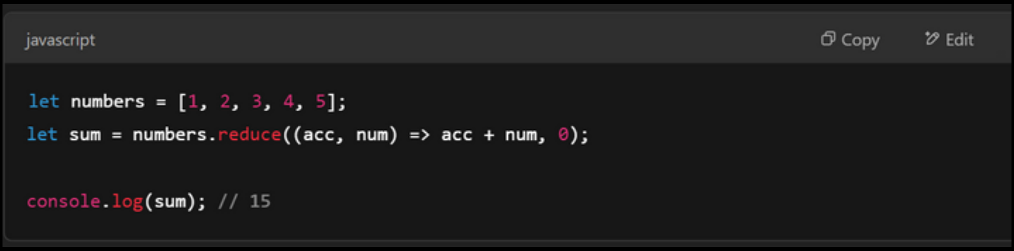
reduce( )

The reduce method in JavaScript is used to accumulate values from an array into a single result. It processes each element and carries forward an accumulated value.

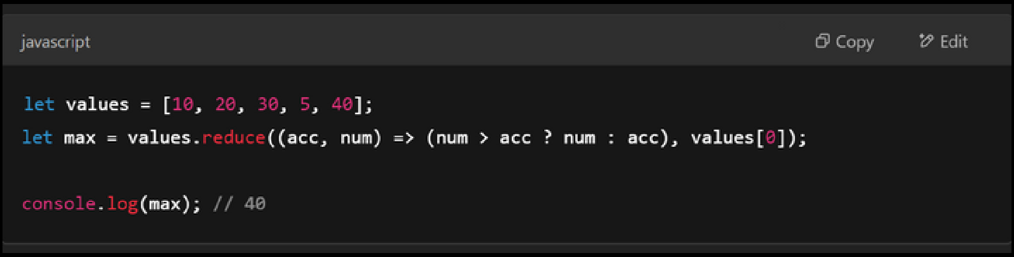


```javascript
array.reduce(callback(accumulator, element, index, array), initialValue);
```

- callback – A function executed for each element.
- accumulator – The accumulated value (previous result).
- element – The current element being processed.
- index (optional) – The index of the current element.
- array (optional) – The original array.
- initialValue (optional) – The starting value of the accumulator.



```javascript
let numbers = [1, 2, 3, 4, 5];
let sum = numbers.reduce((acc, num) => acc + num, 0);

console.log(sum); // 15
```

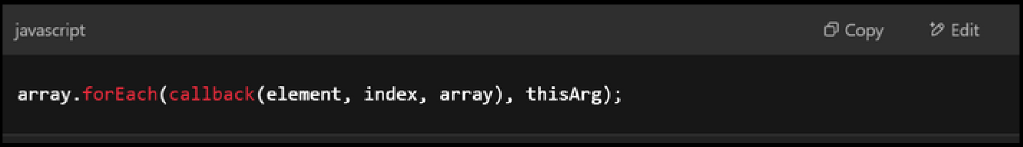


```javascript
let values = [10, 20, 30, 5, 40];
let max = values.reduce((acc, num) => (num > acc ? num : acc), values[0]);

console.log(max); // 40
```

Array Methods

foreach()

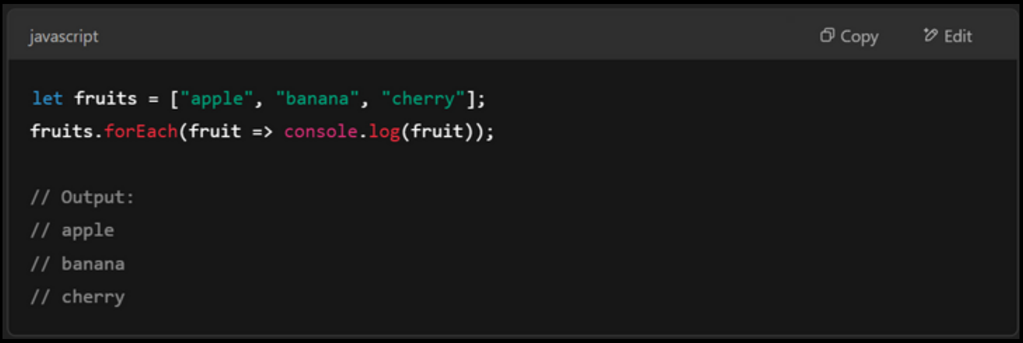
The forEach method in JavaScript is used to iterate over each element in an array and execute a function for each element. It does not return a new array and does not modify the original array unless explicitly done inside the function.



javascript Copy Edit

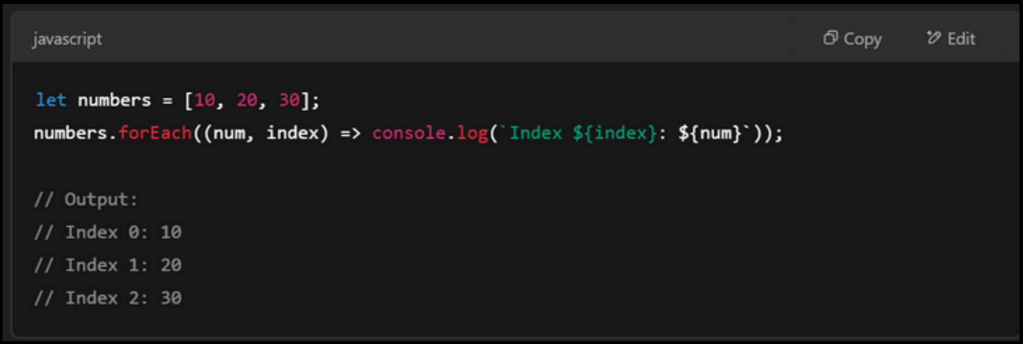
```javascript
array.forEach(callback(element, index, array), thisArg);
```

- callback – A function executed for each element.
- element – The current element being processed.
- index (optional) – The index of the current element.
- array (optional) – The original array.
- thisArg (optional) – Value to use as this in the callback.



javascript Copy Edit

```javascript
let fruits = ["apple", "banana", "cherry"];
fruits.forEach(fruit => console.log(fruit));

// Output:
// apple
// banana
// cherry
```



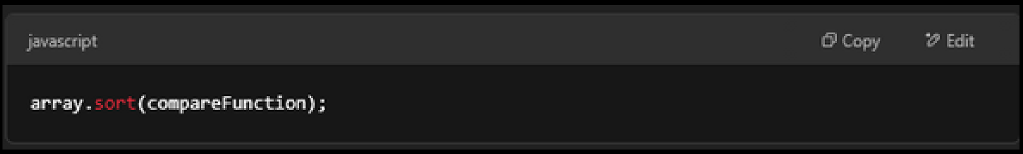
javascript Copy Edit

```javascript
let numbers = [10, 20, 30];
numbers.forEach((num, index) => console.log(`Index ${index}: ${num}`));

// Output:
// Index 0: 10
// Index 1: 20
// Index 2: 30
```

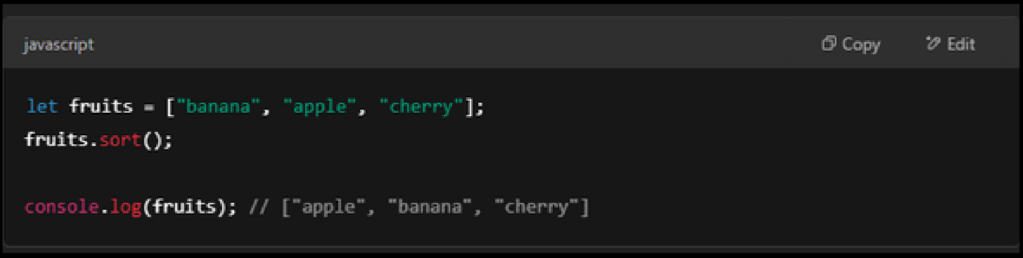
Array Methods

sort( )

The sort method in JavaScript is used to sort elements of an array in place. By default, it sorts elements as strings in ascending order, which can cause unexpected results when sorting numbers.



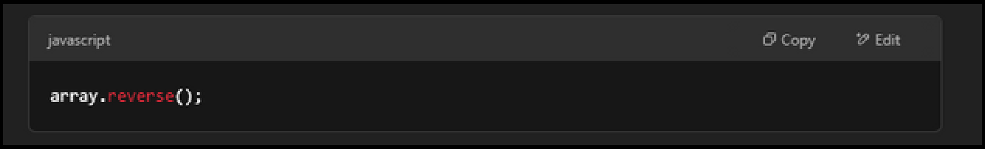
```javascript
array.sort(compareFunction);
```



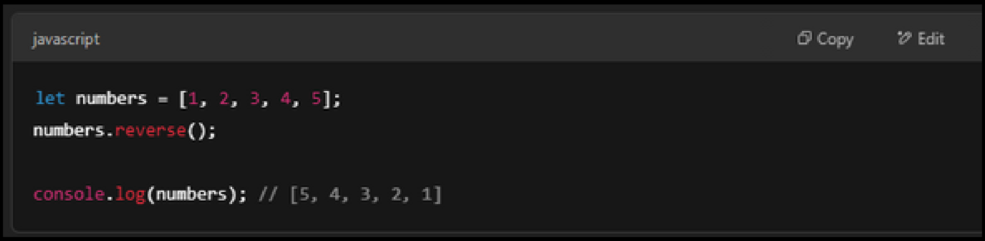
```javascript
let fruits = ["banana", "apple", "cherry"];
fruits.sort();

console.log(fruits); // ["apple", "banana", "cherry"]
```

reverse( )

The reverse method in JavaScript is used to reverse the order of elements in an array. It modifies the original array in place and does not return a new one.



```javascript
array.reverse();
```

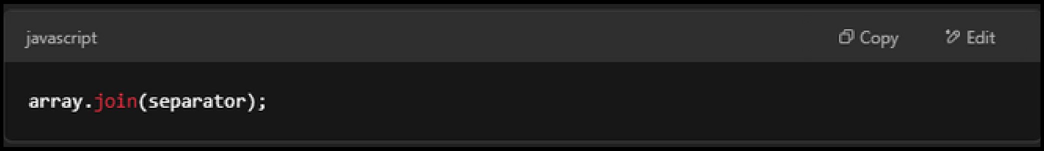


```javascript
let numbers = [1, 2, 3, 4, 5];
numbers.reverse();

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

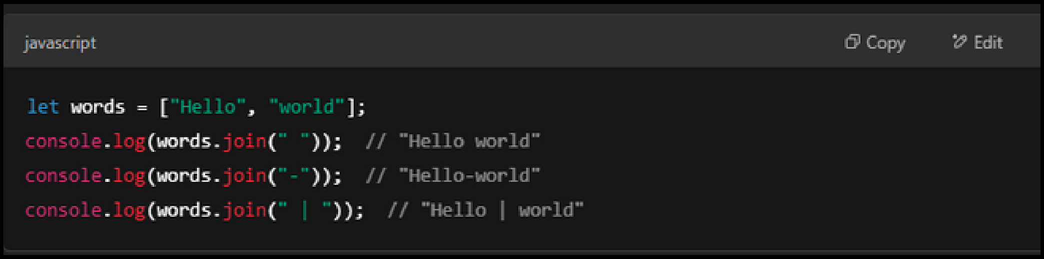
Array Methods

join( )

The join method in JavaScript converts an array into a string, with elements separated by a specified delimiter.



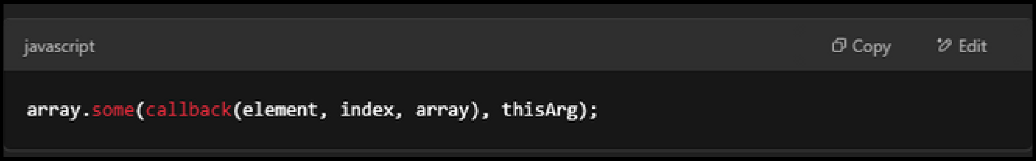
```javascript
array.join(separator);
```



```javascript
let words = ["Hello", "world"];
console.log(words.join(" "));  // "Hello world"
console.log(words.join("-"));  // "Hello-world"
console.log(words.join(" | "));  // "Hello | world"
```

some()

The some method in JavaScript is used to check if at least one element in an array satisfies a given condition. It returns true or false.



```javascript
array.some(callback(element, index, array), thisArg);
```