JavaScript Array Methods

Haris Ahmad

Save for Later

Array Methods...

1. values()
2. length()
3. reverse()
4. sort()
5. at()
6. fill()
7. from()
8. join()
9. toString()
10. pop()
11. forEach()
12. shift()
13. copyWithin()
14. push()
15. unshift()
16. concat()
17. some()
18. splice()
19. flat()
20. lastIndexOf()
21. of()
22. every()
23. slice()
24. flatMap()
25. findIndex()
26. find()
27. inculdes()
28. entries()
29. reduceRight()
30. reduce()
31. isArray()
32. filter()
33. keys()
34. map()

JS

values(): This method returns an iterator that provides the values for each index in the array. It takes no arguments.

```
const arr = ['apple', 'banana', 'cherry']
const iterator = arr.values();

for (const value of iterator) {
  console.log(value);
} // Output: apple banana cherry
```

length(): This property returns the length of the array.

```
const arr = ['apple', 'banana', 'cherry']
console.log(arr.length); // Output: 3
```

Haris Ahmad

Save for Later

reverse(): This method reverses the order of the elements in the array.

```
const arr = ['apple', 'banana', 'cherry'];
arr.reverse();
console.log(arr); // Output: ['cherry', 'banana', 'apple']
```

sort(): This method sorts the elements of an array in place and returns the sorted array. It can take an optional compare function as an argument.

```
const arr = ['banana', 'apple', 'cherry'];
arr.sort();
console.log(arr); // Output: ['apple', 'banana', 'cherry']
```

at(): This method returns the element at the specified index in the array. It takes one argument: the index.

```
const arr = ['apple', 'banana', 'cherry'];
console.log(arr.at(1)); // Output: 'banana'
```

fill(): This method fills all the elements of an array from a start index to an end index with a static value. It can take up to three arguments: the value to fill with, the start index, and the end index.

```
const arr = ['apple', 'banana', 'cherry'];
arr.fill('orange', 1, 2);
console.log(arr); // Output: ['apple', 'orange', 'cherry']
```

Haris Ahmad

Save for Later

from(): This method creates a new array from an array-like object or an iterable object. It can take up to two arguments: the object to convert to an array, and a mapping function to apply to each element of the new array.

```javascript
const obj = { 0: 'apple', 1: 'banana', 2: 'cherry', length: 3 }
const arr = Array.from(obj);
console.log(arr); // Output: ['apple', 'banana', 'cherry']
```

join(): This method joins all the elements of an array into a string using a specified separator. It takes one optional argument: the separator to use.

```javascript
const arr = ['apple', 'banana', 'cherry'];
const str = arr.join(', ');
console.log(str); // Output: 'apple, banana, cherry'
```

>>>

toString(): This method returns a string representing the array and its elements.

```
const arr = ['apple', 'banana', 'cherry'];
const str = arr.toString();
console.log(str); // Output: 'apple,banana,cherry'
```

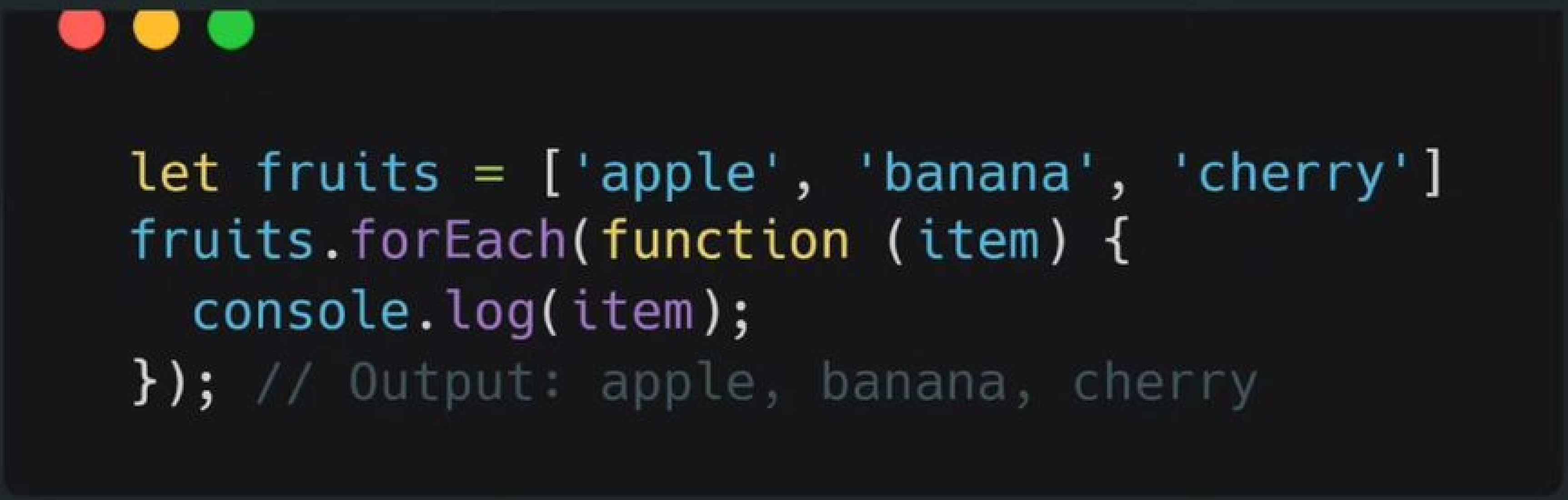
pop(): This method removes the last element from an array and returns that element.

```
const arr = ['apple', 'banana', 'cherry'];
const last = arr.pop();
console.log(last); // Output: 'cherry'
console.log(arr); // Output: ['apple', 'banana']
```

Haris Ahmad

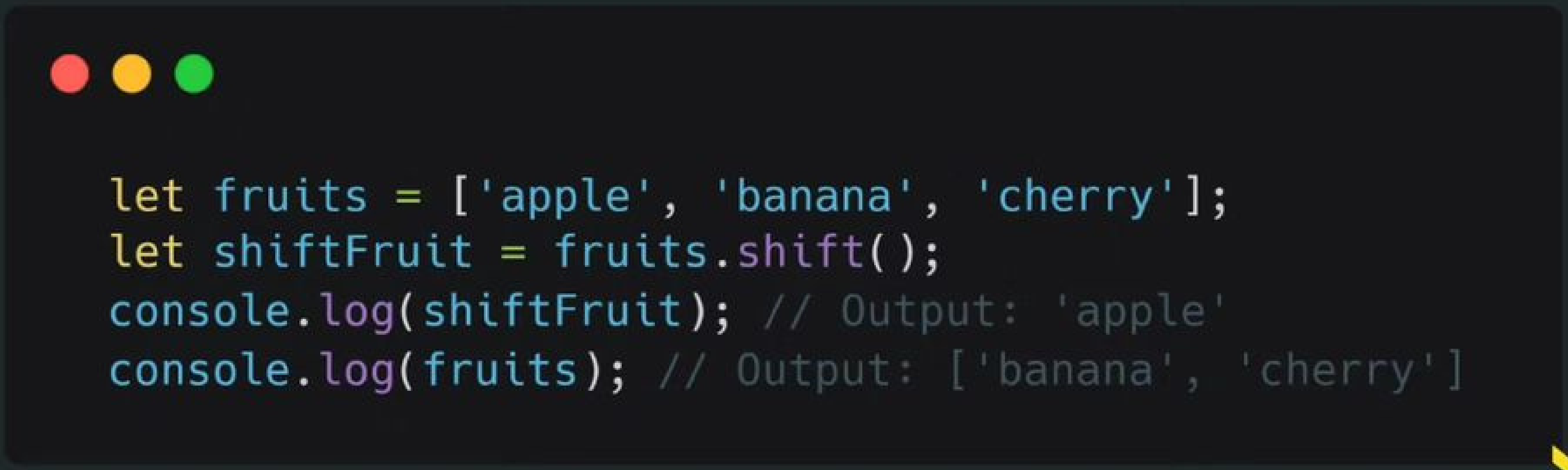
Save for Later

forEach() method executes a provided function once for each array element. It doesn't return anything, it just executes the callback function on each element of the array.



```
let fruits = ['apple', 'banana', 'cherry']
fruits.forEach(function (item) {
  console.log(item);
}); // Output: apple, banana, cherry
```

shift() method removes the first element from an array and returns that removed element. This method changes the length of the array.



```
let fruits = ['apple', 'banana', 'cherry'];
let shiftFruit = fruits.shift();
console.log(shiftFruit); // Output: 'apple'
console.log(fruits); // Output: ['banana', 'cherry']
```

copyWithin() method shallow copies part of an array to another location in the same array and returns the modified array without modifying its length.**Syntax** .copyWithin(target, start, end)

```
let numbers = [1, 2, 3, 4, 5];
numbers.copyWithin(2, 0, 2);
console.log(numbers); // Output: [1, 2, 1, 2, 5]
```

push() method adds one or more elements to the end of an array and returns the new length of the array.

```
let fruits = ['apple', 'banana'];
fruits.push('cherry', 'orange');
console.log(fruits); // Output: ['apple', 'banana', 'cherry', 'orange']
```

unshift() method adds one or more elements to the beginning of an array and returns the new length of the array.

```javascript
let fruits = ['cherry', 'orange'];
fruits.unshift('apple', 'banana');
console.log(fruits); // Output: ['apple', 'banana', 'cherry', 'orange']
```

concat() method is used to merge two or more arrays. This method does not change the existing arrays, but instead returns a new array.

```javascript
let fruits = ['apple', 'banana'];
let moreFruits = ['cherry', 'orange'];
let allFruits = fruits.concat(moreFruits);
console.log(allFruits); // Output: ['apple', 'banana', 'cherry', 'orange']
```

>>>

CodeHype

Save for Later

splice() method changes the contents of an array by removing or replacing existing elements and/or adding new elements in place.

```
const fruits = ['apple', 'banana', 'cherry', 'orange'];
//Syntax : arr.splice(start, deleteCount, item1, ..., itemN)
fruits.splice(2, 1, 'mango', 'kiwi');
console.log(fruits); // Output: [ 'apple', 'banana', 'mango', 'kiwi', 'orange'
]
```

flat() This method creates a new array with all sub-array elements concatenated into it recursively up to the specified depth.

```
const numbers = [1, [2, [3]], 4];
const flatNumbers = numbers.flat(Infinity);
console.log(flatNumbers); // Output: [1, 2, 3, 4]
```

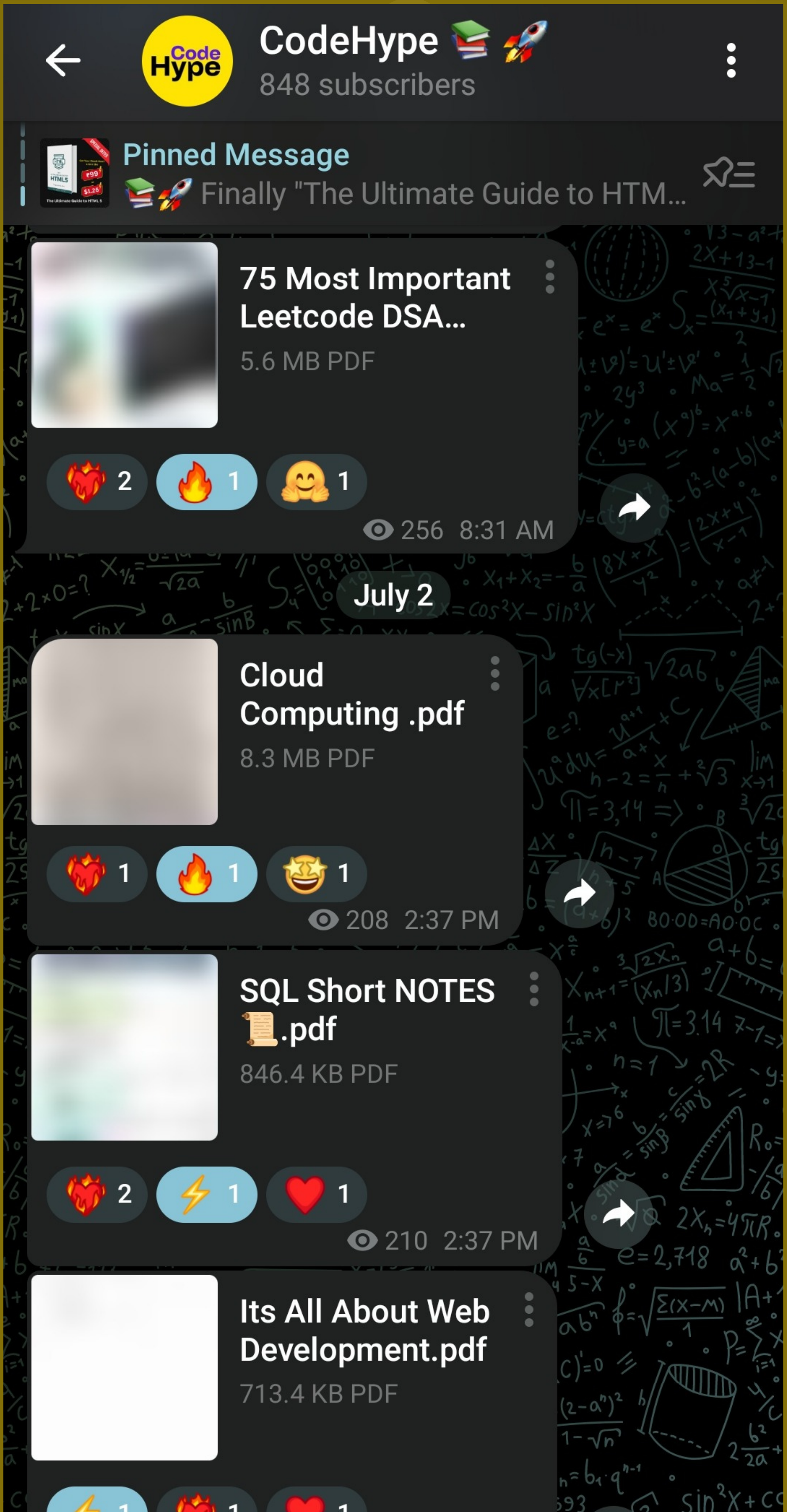
>>>

Follow for more useful content

Telegram
@codehype

Join for FREE
coding resources

CodeHype


CodeHype 📚🚀
848 subscribers

Pinned Message
📚🚀 Finally "The Ultimate Guide to HTM...

75 Most Important Leetcode DSA...
5.6 MB PDF
❤️‍🔥 2 🔥 1 🤗 1
256 8:31 AM

July 2

Cloud Computing .pdf
8.3 MB PDF
❤️‍🔥 1 🔥 1 🤩 1
208 2:37 PM

SQL Short NOTES 📜.pdf
846.4 KB PDF
❤️‍🔥 2 ⚡ 1 ❤️ 1
210 2:37 PM

Its All About Web Development.pdf
713.4 KB PDF