Contents

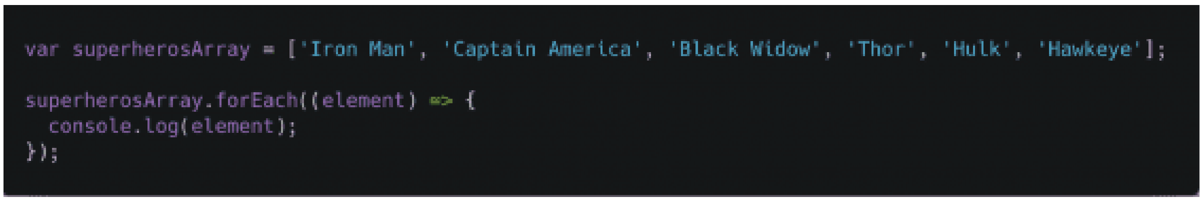
1.0 Most common Array Iterators

1.1 Array.forEach()

Method name: forEach
Returns: undefined

This method executes a snippet of code (or a function) once for every element of an array.

For example:



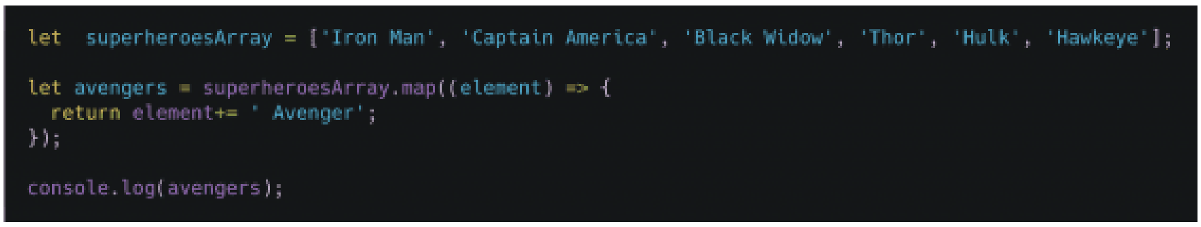
```
var superherosArray = ['Iron Man', 'Captain America', 'Black Widow', 'Thor', 'Hulk', 'Hawkeye'];

superherosArray.forEach((element) => {
  console.log(element);
});
```

The forEach method is called for the superheroes array. The argument of forEach() method is a *callback* function. This function is executed for every element of the array. Each element is passed as an argument to this callback function.

1.2 Array.map()

Method Name: map
Returns: A new array

This method **returns a new array** with the updated elements after calling a callback function on every element in the array.



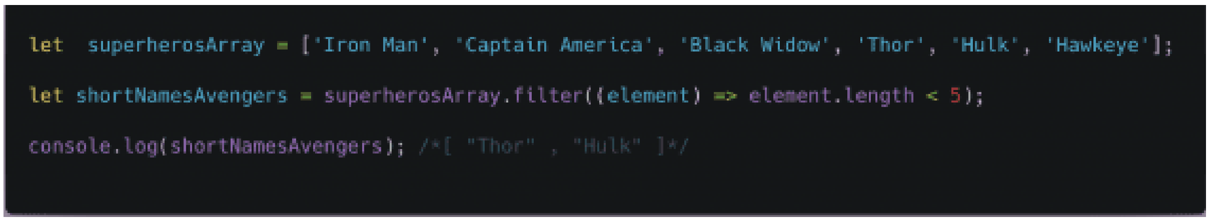
```
let  superheroesArray = ['Iron Man', 'Captain America', 'Black Widow', 'Thor', 'Hulk', 'Hawkeye'];

let avengers = superheroesArray.map((element) => {
  return element+= ' Avenger';
});

console.log(avengers);
```

The map method is called on the superheroes array. This method has as an argument a callback function. Map returns a *new* array, which has the string 'Avengers' concatenated in the original values! The original array doesn't change.

1.3 Array.filter()

Method name: filter
Returns: A new array

This method checks each element in an array to see if it meets a condition. It returns a new array with the elements that meet the condition.



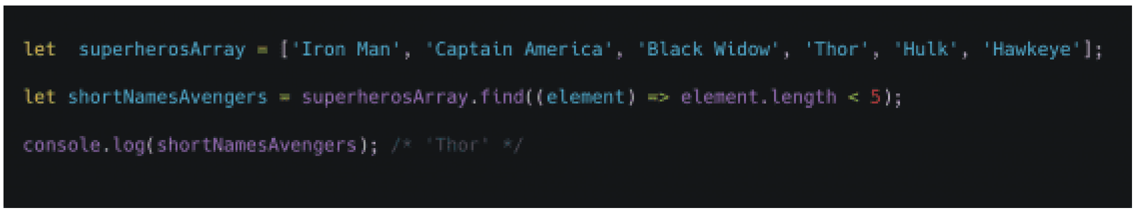
```
let  superherosArray = ['Iron Man', 'Captain America', 'Black Widow', 'Thor', 'Hulk', 'Hawkeye'];

let shortNamesAvengers = superherosArray.filter((element) => element.length < 5);

console.log(shortNamesAvengers); /*[ "Thor" , "Hulk" ]*/
```

The callback function for the .filter() method should return true or false depending on if the element length is shorter than 5.
The elements that cause the callback function to return true are added to the new array.

1.4 Array.find()

Method name: find
Returns: The value of the first occurrence of the element, undefined if the element doesn't exist

This method returns the **value** of the *first* element of an array which satisfies a condition. The method will return *undefined* if none of the elements satisfies this condition.

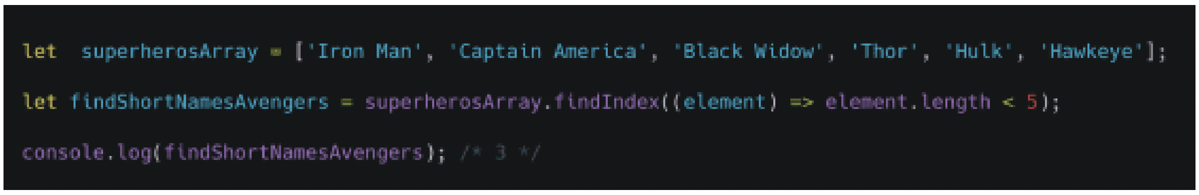


```
let  superherosArray = ['Iron Man', 'Captain America', 'Black Widow', 'Thor', 'Hulk', 'Hawkeye'];

let shortNamesAvengers = superherosArray.find((element) => element.length < 5);

console.log(shortNamesAvengers); /* 'Thor' */
```

1.5 Array.findIndex()

Method name: findIndex
Returns: The index of the first occurrence of the element, -1 if the element doesn't exist

Similar to find method. Their difference is that this method returns the **index** of the first element of an array which satisfies the condition set. The method will return *-1* if none of the elements satisfies the condition.

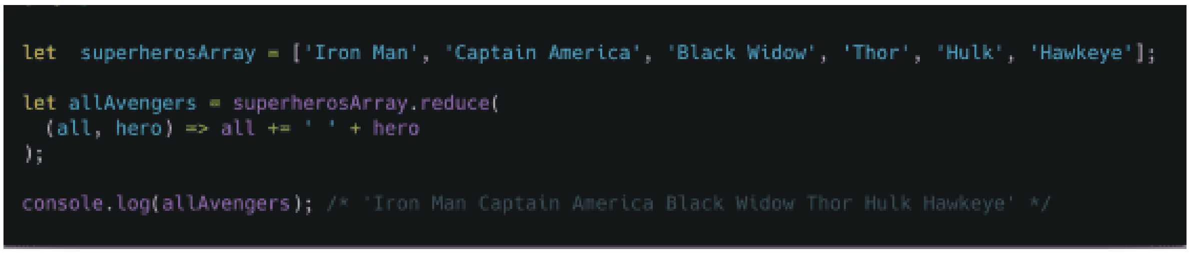


```
let  superherosArray = ['Iron Man', 'Captain America', 'Black Widow', 'Thor', 'Hulk', 'Hawkeye'];

let findShortNamesAvengers = superherosArray.findIndex((element) => element.length < 5);

console.log(findShortNamesAvengers); /* 3 */
```

1.6 Array.reduce()

Method name: reduce
Returns: A single value

The reduce method is used to reduce the array to a single value. It executes a provided function for each value of the array (from left-to-right). The return value of the function is stored in an accumulator.



```
let  superherosArray = ['Iron Man', 'Captain America', 'Black Widow', 'Thor', 'Hulk', 'Hawkeye'];

let allAvengers = superherosArray.reduce(
  (all, hero) => all += ' ' + hero
);

console.log(allAvengers); /* 'Iron Man Captain America Black Widow Thor Hulk Hawkeye' */
```

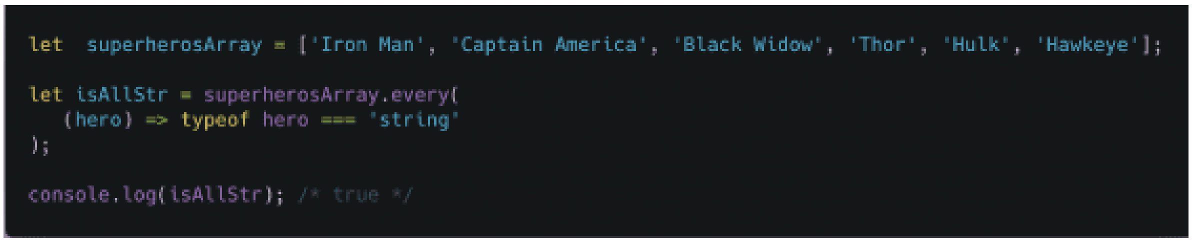
In this example, Reduce accepts two parameters, the accumulator (all) and the current element (hero). The reduce method iterates through each element in the array as a for-loop. In the accumulator, we store the concatenated string.

1.7 Array.every()

Method name: every
Returns: boolean

--

The 'every' method tests if all elements in the array pass a condition. The return value is a boolean.

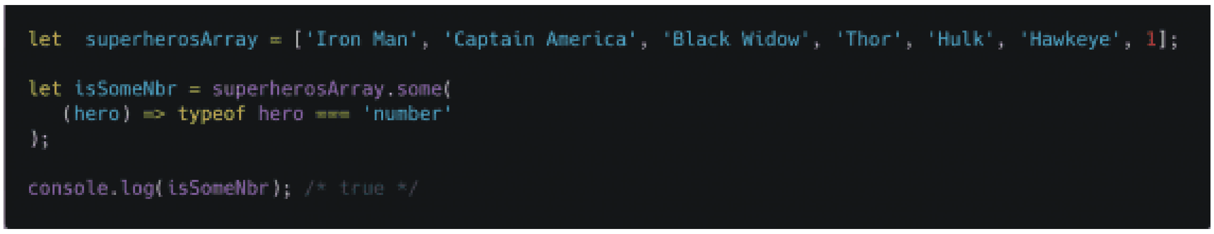


```
let  superherosArray = ['Iron Man', 'Captain America', 'Black Widow', 'Thor', 'Hulk', 'Hawkeye'];

let isAllStr = superherosArray.every(
   (hero) => typeof hero === 'string'
);

console.log(isAllStr); /* true */
```

1.8 Array.some()

Method name: some
Returns: boolean

The 'some' method tests if some of the elements in the array pass a condition. The return value is a boolean.



```
let  superherosArray = ['Iron Man', 'Captain America', 'Black Widow', 'Thor', 'Hulk', 'Hawkeye', 1];

let isSomeNbr = superherosArray.some(
   (hero) => typeof hero === 'number'
);

console.log(isSomeNbr); /* true */
```

2.0 JavaScript Arrays Cheat Sheet

2.1 Static Properties

```
Array.from('123'); // ['1','2','3']
Array.isArray([1,2,3]); // true
Array.of(1,2,3) // [1,2,3]
```

2.2 Instance Properties

Search or run test on array

```
[1,2,2,3].indexOf(2); // 1
[1,2,2,3].lastIndexOf(2); // 2
[1,2,2,3].filter(n => n === 2); // [2,2]
[1,2,2,3].find(n => n === 2); // 2
[1,2,2,3].findIndex(n => n % 2 === 0); // 1
[1,2,2,3].every(n => n % 2 === 0); // false
[1,2,2,3].some(n => n % 2 === 0); // true
[1,2,2,3].includes(4); // false
```

Loop through array

```
for (const value of [4,5,6].values())
  console.log(value); // 4 → 5 → 6
for (const [i,n] of [4,5,6].entries())
  console.log(i,n); // 0 4 → 1 5 → 2 6
[4,5,6].forEach((n,i) => console.log(i,n)); // 0 4 → 1 5 → 2 6
[4,5,6].reduce((acc,cur) => acc + cur, 0); // 15
[4,5,6].map(n => n + 1); // [5,6,7]
[4,5,6].flatMap(n => [n + 1]); // [5,6,7]
[4,5,6].flatMap(n => [[n + 1]]); // [[5],[6],[7]]
```

Return new array

```
[1,2,3].concat([4]); // [1,2,3,4]
[1,2,3].join(' + '); // '1 + 2 + 3'
[1,2,3].slice(1,2); // [2]
[1,2,3].slice(); // [1,2,3]
[1,2,3].toString(); // '1,2,3'
[1,2,[3,4]].flat(); // [1,2,3,4]
[1,2,[[3,4]]].flat(1); // [1,2,[3,4]]
[1,2,[[3,4]]].flat(2); // [1,2,3,4]
```

Modify original array

```
[1,2,3,4].copyWithin(2,0); // [1,2,1,2]
[1,2,3].splice(1,2); // [2,3]
[1,2,3].reverse(); // [3,2,1]
[1,2,3].fill(4); // [4,4,4]
[1,2,3].pop(); // [1,2]
[1,2,3].push(4); // [1,2,3,4]
[1,2,3].shift(); // 1
[1,2,3].unshift(4); // 4
[2,3,1].sort(); // [1,2,3]
```

2.3 Explanations

In these examples, `i` refers to an index.

- `Array.from(iter)` creates array from an iterable object
- `Array.isArray(arr)` checks for an array
- `Array.of(arr)` creates a new array with provided array
- `[].indexOf(elem)` returns index of first instance of `elem` in array, or `-1`
- `[].lastIndexOf(elem)` returns index of last instance of `elem` in array, or `-1`
- `[].filter(fn)` returns array of elements that satisfy our `fn` test
- `[].find(fn)` returns element in the array that satisfies our `fn` test, or `undefined`
- `[].findIndex(fn)` returns index of the first element that satisfies our `fn` test, or `-1`
- `[].every(fn)` checks if every element in the array satisfies our `fn` test
- `[].some(fn)` checks if at least one element in the array satisfies our `fn` test
- `[].includes(elem)` checks if array contains `elem`
- `[].values()` returns an iterator to loop over elements in array
- `[].entries()` returns an iterator to loop over index/element pairs in array
- `[].forEach(fn)` executes `fn` once for each element
- `[].reduce(fn)` reduce values of an array to a single value
- `[].map(fn)` creates new array by calling a `fn` for each element
- `[].flatMap()` runs `map()` followed by `flat(1)`
- `[].concat(arr)` joins 2 arrays
- `[].join(str)` joins all elements of an array into a string delimited by `str`
- `[].slice(i1,i2)` returns new array from `[i1, i1+i2)`
- `[].slice()` copies an array
- `[].toString()` converts array to string
- `[].flat(n)` flattens all sub-array elements up to a specified, zero-indexed depth `n`
- `[].copyWithin(i1, i2)` copies sequence of elements from `i2` to the end to index `i1` onward
- `[].splice(i,n)` starting from `i`, removes `n` elements and returns the modified array
- `[].reverse()` reverses order of the array
- `[].fill(elem)` fills all elements with `elem`
- `[].pop()` removes and returns last element of array
- `[].push(elem)` adds `elem` to end of array and returns length
- `[].shift()` removes and returns first element of array
- `[].unshift(elem)` adds `elem` to beginning of array and returns length
- `[].sort()` sorts an array