8 Must-Know JavaScript Array Methods

every

Syntax

```
Array.prototype.every(callback)
```

What does it do?

It tests whether all elements in the array pass the test implemented by the provided function. It returns a Boolean value.

```
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]

const allAreOdd = numbers.every(number => {
    return number % 2 !== 0
})

console.log(allAreOdd)

// Output: false
```

find

Syntax

```
Array.prototype.find(callback)
```

What does it do?

It returns the first element in the provided array that satisfies the provided testing function. If no values satisfy the testing function, `undefined` is returned.

```
const numbers = [2, 4, 5, 7, 10]

const firstOddNumber = numbers.find(number => {
    return number % 2 !== 0
})

console.log(firstOddNumber)

// Output: 5
```

filter

Syntax

```
Array.prototype.filter(callback)
```

What does it do?

It creates a new array with all elements that pass the test implemented by the provided function.

```
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]

const onlyOddNumbers = numbers.filter(number => {
    return number % 2 !== 0
})

console.log(onlyOddNumbers)

// Output: [1, 3, 5, 7, 9]
```

forEach

Syntax

```
Array.prototype.forEach(callback)
```

What does it do?

It executes a provided function once for each array element.

```
const names = ['John', 'Jane']

names.forEach(name => {
    console.log('I am ' + name)
})

// Output: I am John
           I am Jane
```

includes

Syntax

```
Array.prototype.includes(searchElement)
```

What does it do?

It determines whether an array includes a certain value among its entries, returning `true` or `false` as appropriate.

```
const animals = [‘Lion’, ‘Dog’, ‘Cat’]

const isGiraffeHere = animals.includes(‘Giraffe’)

console.log(isGiraffeHere)

// Output: false
```

some

Syntax

```
Array.prototype.some(callback)
```

What does it do?

It tests whether at least one element in the array passes the test implemented by the provided function.

```
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]

const atLeastOneIsOdd = numbers.some(number => {
    return number % 2 !== 0
})

console.log(atLeastOneIsOdd)

// Output: true
```

reduce

Syntax

```
Array.prototype.reduce(callback)
```

What does it do?

It executes a provided function on each element of the array, in order, passing in the return value from the calculation on the preceding element. The final result is a single value.

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

const total = numbers.reduce((total, amount) => {
    return total + amount
})

console.log(total)

// Output: 15
```

map

Syntax

```
Array.prototype.map(callback)
```

What does it do?

It creates a new array populated with the results of calling a provided function on every element in the calling array.

```
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]

const squared = numbers.map(number => {
    return number * number
})

console.log(squared)

// Output: [1, 4, 9, 16, 25, 36, 49, 64, 81]
```

FOLLOW