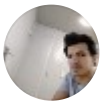
Published in JavaScript in Plain English



Selahattindemir Follow

Dec 16, 2021 · 4 min read · Listen

A Guide to React Native Array Methods

Let's consider what an array method is. Simply put, React array methods are the methods that are available as array prototypes with the new version of JavaScript, by which we can navigate the array elements one by one with the help of a callback function and perform certain checks

First, keep in mind that this article most likely applies to other programming languages and frameworks you use, as these are concepts that exist in many other languages and frameworks.

As an example, let's say we have data like this.

```
  { id: 2, name: "Notebook", price: 10 },
  { id: 3, name: "Eraser", price: 2 },
  { id: 4, name: "Sharpener", price: 7 },
];
```

.find()

Used to find an element in an array. As soon as the element is found, the search stops and returns the found element. If there is another element that meets the same conditions, the first element found is returned.

```
products.find((product) => product.price > 5); // {id: 2, name:
"Notebook", price: 10}
```

.some()

Returns true/false whether at least one element in the array satisfies the given condition.

```
products.some((product) => product.price < 2); // false, There isn't
a single one in the array t                    than 2
products.some((product) => p                   > 9); // true, There is at
least one price greater than 9 in the array
```


.every()

Returns true/false whether all elements in the array meet the entered condition.

```
products.every((product) => product.price > 1); // true, All product
prices are greater than 1
products.every((product) => product.price < 10); // false, All item
prices NOT less than 10
```

.includes()

```
products.every((product) => product.name.includes("Pencil")); //
false, All product names do not contain Pencil
```

.map()

It applies the entered callback function to the given array elements and creates a new array with the results obtained. Simply put, it helps us create a new set of array based on an existing one.

Keep in mind that the resulting array will always be the same length as the original array.

```
products.map((product) => `${product.name} price ${product.price}
was dollars.`);
// ["The price of the pencil is $5.", "The price of the notebook is
$10.", "The price of the eraser is $2.", "The price of the sharpener
is $7."]
```

In React Native:

```
function ShowProducts({ productList }) {
  return productList.map((product) => <Text key={product.id}>
{product.name}</Text>);
}
```

.filter()

Just as it sounds, it works similar to the way the `.map()` method works. Applies the entered callback function to the given array elements. Eliminates results returning false and creates a new array with results returning true.

```
products.filter((product) => product.name.includes("Pencil")); //
[{id: 1, name: "Pencil", price: 5}, {id: 4, name: "Sharpener",
price: 7}]
```


```
products
  .filter((product) => product.name.includes("Pencil")) // Find
products with Pencil first
  .map((product) => `${product.name} price ${product.price} was
dollars.`); // Then apply the given callback function to the found
elements

// ["The price of the pencil is $5.", "The price of the sharpener is
$7."]
```

.reduce()

It applies the callback function given as reducer on the elements of the given array. The result returned by this function is remembered in each loop and the next returned result is added to the previous one. The reducer function takes four parameters: accumulator (the sum of the result from each loop), current value (value of the next value in the array), current index (position of the next value in the array), and source array (array on which reduce is applied).

`.reduce()` it can also take an initial value as a second parameter. For mathematical operations, a number will be entered as a start, as well as string, array, etc. can also be entered.

The total price of the products in the list:

```
products.reduce((total, product) => total + product.price, 0); // 24
```

Writing product names in one line:

```
products.reduce((names, product) => names + " " + product.name,
"Product Names:"); // "Product Names: Pencil Notebook Eraser
Sharpener"
```

```
    ...newProductList,
    `${product.name} ${product.fiyat} was dollars.`,
  ],
  [] // Initial value is empty array
);
// ["The price of the pencil is $5.", "The price of the notebook is
$10.", "The price of the eraser is $2.", "The price of the sharpener
is $7."]
```

If you found this article interesting do clap for me with the clap icon.

Keep coding!

*More content at **plainenglish.io**. Sign up for our **free weekly newsletter here**.*