**It's Just JavaScript**

One of the great things about React is that a lot of what you'll be using is regular JavaScript. To make sure you're ready to move forward, please take a look at the following code:

**const** shelf1 = [{name: 'name1', shelf: 'a'},{name: 'name2', shelf: 'a'}];

**const** shelf2 = [{name: 'name3', shelf: 'b'},{name: 'name4', shelf: 'b'}];

**const** allBooks = [...shelf1, ...shelf2];

**const** filter = books => shelf => books.filter(b => {

**return** b.shelf === shelf;

});

**const** filterBy = filter(allBooks);

**const** booksOnShelf = filterBy('b');

If *any* of the code above looks confusing, or if you simply need a refresher on E6, please go through [our ES6 course](https://classroom.udacity.com/courses/ud356) before moving forward.

Over the past couple of years, functional programming has had a large impact on the JavaScript ecosystem and community. Functional programming is an advanced topic in JavaScript and fills hundreds of books. It's too complex to delve into the benefits of functional programming (we've got to get to React content, right?!?). But React builds on a lot of the techniques of functional programming...techniques that you'll learn as you go through this program. However, there are a couple of important JavaScript functions that are vital to functional programming that we should look at. These are the Array's .map() and .filter() methods.

**Array's .map() Method**

If you're not familiar with JavaScript's [Array .map() method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map), it gets called on an existing array and returns a new array based on what is returned from the function that's passed as an argument. Let's take a look:

**const** names = ['Karen', 'Richard', 'Tyler'];

**const** nameLengths = names.map( name => name.length );

Let's go over what's happening here. The .map() method works on arrays, so we have to have an array to start with:

**const** names = ['Karen', 'Richard', 'Tyler'];

We call .map() on the names array and pass it a function as an argument:

names.map( name => name.length );

The arrow function that's passed to .map() gets called *for each item* in the names array! The arrow function receives the first name in the array, stores it in the name variable and returns its length. Then it does that again for the remaining two names.

.map() returns a new array with the values that are returned from the arrow function:

**const** nameLengths = names.map( name => name.length );

So nameLengths will be a *new* array [5, 7, 5]. This is important to understand; **the .map() method returns a new array, it does not modify the original array**.

This was just a brief overview of how the .map() method works. For a deeper dive, check out [.map() on MDN](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map).

**.map() Quiz**

Use the provided music data array and the .map() method to create a new array that contains items in the format:

<album-name> by <artist> sold <sales> copies

Store the new array in an albumSalesStrings array. So the first item in the albumSalesStrings array should be "25 by Adele sold 1731000 copies"





\* Using the musicData array and .map():

\* - return a string for each item in the array in the following format

\* <album-name> by <artist> sold <sales> copies

\* - store the returned data in a new albumSalesStrings variable

\*

\* Note:

\* - do not delete the musicData variable

\* - do not alter any of the musicData content

\* - do not format the sales number, leave it as a long string of digits

\*/

const musicData = [

{ artist: 'Adele', name: '25', sales: 1731000 },

{ artist: 'Drake', name: 'Views', sales: 1608000 },

{ artist: 'Beyonce', name: 'Lemonade', sales: 1554000 },

{ artist: 'Chris Stapleton', name: 'Traveller', sales: 1085000 },

{ artist: 'Pentatonix', name: 'A Pentatonix Christmas', sales: 904000 },

{ artist: 'Original Broadway Cast Recording',

name: 'Hamilton: An American Musical', sales: 820000 },

{ artist: 'Twenty One Pilots', name: 'Blurryface', sales: 738000 },

{ artist: 'Prince', name: 'The Very Best of Prince', sales: 668000 },

{ artist: 'Rihanna', name: 'Anti', sales: 603000 },

{ artist: 'Justin Bieber', name: 'Purpose', sales: 554000 }

];

const albumSalesStrings = musicData.map(album => `${album.name} by ${album.artist} sold ${album.sales} copies`);

console.log(albumSalesStrings);

* RESET QUIZ
* TEST RUN
* SUBMIT ANSWER

**Map Quiz Solution Code**

Once you've tried your hand at solving this quiz, hover your mouse here for one possible solution.

**Array's .filter() Method**

JavaScript's [Array .filter() method](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter) is similar to the .map() method:

* it is called on an array
* it takes a function as an argument
* it returns a new array

The difference is that the function passed to .filter() is used as a test, and only items in the array that pass the test are included in the new array. Let's take a look at an example:

**const** names = ['Karen', 'Richard', 'Tyler'];

**const** shortNames = names.filter( name => name.length < 6 );

Just as before, we have the starting array:

**const** names = ['Karen', 'Richard', 'Tyler'];

We call .filter() on the names array and pass it a function as an argument:

names.filter( name => name.length < 6 );

Again, just like with .map() the arrow function that's passed to .filter() gets called *for each item* in the names array. The first item (i.e. 'Karen') is stored in the name variable. Then the test is performed - this is what's doing the actual filtering. It checks the length of the name. If it's 6 or greater, then it's skipped (and *not* included in the new array!). But if the length of the name is less than 6, then name.length < 6 returns true and the name \_is\_ included in the new array!

And lastly, just like with .map() the .filter() method returns a *new* array instead of modifying the original array:

**const** shortNames = names.filter( name => name.length < 6 );

So shortNames will be the new array ['Karen', 'Tyler'].

Notice that it only has two names in it now, because 'Richard' is 7 characters and was filtered out.

This was just a brief overview of how the .filter() method works. For a deeper dive, check out [.filter() on MDN](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter).

**.filter() Quiz**

Use the provided music data array and the .filter() method to create a new array that only contains albums with names between 10 and 25 characters long. Store the new array in a variable called results.





/\* Using .filter()

\*

\* Using the musicData array and .filter():

\* - return only album objects where the album's name is

\* 10 characters long, 25 characters long, or anywhere in between

\* - store the returned data in a new `results` variable

\*

\* Note:

\* - do not delete the musicData variable

\* - do not alter any of the musicData content

\*/

const musicData = [

{ artist: 'Adele', name: '25', sales: 1731000 },

{ artist: 'Drake', name: 'Views', sales: 1608000 },

{ artist: 'Beyonce', name: 'Lemonade', sales: 1554000 },

{ artist: 'Chris Stapleton', name: 'Traveller', sales: 1085000 },

{ artist: 'Pentatonix', name: 'A Pentatonix Christmas', sales: 904000 },

{ artist: 'Original Broadway Cast Recording',

name: 'Hamilton: An American Musical', sales: 820000 },

{ artist: 'Twenty One Pilots', name: 'Blurryface', sales: 738000 },

{ artist: 'Prince', name: 'The Very Best of Prince', sales: 668000 },

{ artist: 'Rihanna', name: 'Anti', sales: 603000 },

{ artist: 'Justin Bieber', name: 'Purpose', sales: 554000 }

];

const results = musicData.filter(album => album.name.length <=25 && album.name.length >=10);

console.log(results);

* RESET QUIZ
* TEST RUN
* SUBMIT ANSWER

**Filter Quiz Solution Code**

Once you've tried your hand at solving this quiz, hover your mouse here for one possible solution.

**Combining .map() And .filter() Together**

What makes .map() and .filter() so powerful is that they can be combined. Because both methods return arrays, we can chain the method calls together so that the returned data from one can be a new array for the next.

**const** names = ['Karen', 'Richard', 'Tyler'];

**const** shortNamesLengths = names.filter( name => name.length < 6 ).map( name => name.length );

To break it down, the names array is filtered, which returns a new array, but then .map() is called on that new array, and returns a new array of its own! This new array that's returned from .map() is what's stored in shortNamesLengths.

**.filter() First!**

On a side note, you'll want to run things in this order (.filter() first and then .map()). Because .map() runs the function once *for each item* in the array, it will be faster if the array were already filtered.

**.filter() and .map() Quiz**

Using the same music data, use .filter() and .map() to filter and map over the list and store the result in a variable named popular. Use .filter() to filter the list down to just the albums that have sold over 1,000,000 copies. Then chain .map() onto the returned array to create a new array that contains items in the format:

<artist> is a great performer

The first item in the popular array will be 'Adele is a great performer'.



\* - filter the musicData array down to just the albums that have

\* sold over 1,000,000 copies

\* - on the array returned from .filter(), call .map()

\* - use .map() to return a string for each item in the array in the

\* following format: "<artist> is a great performer"

\* - store the array returned form .map() in a new "popular" variable

\*

\* Note:

\* - do not delete the musicData variable

\* - do not alter any of the musicData content

\*/

const musicData = [

{ artist: 'Adele', name: '25', sales: 1731000 },

{ artist: 'Drake', name: 'Views', sales: 1608000 },

{ artist: 'Beyonce', name: 'Lemonade', sales: 1554000 },

{ artist: 'Chris Stapleton', name: 'Traveller', sales: 1085000 },

{ artist: 'Pentatonix', name: 'A Pentatonix Christmas', sales: 904000 },

{ artist: 'Original Broadway Cast Recording',

name: 'Hamilton: An American Musical', sales: 820000 },

{ artist: 'Twenty One Pilots', name: 'Blurryface', sales: 738000 },

{ artist: 'Prince', name: 'The Very Best of Prince', sales: 668000 },

{ artist: 'Rihanna', name: 'Anti', sales: 603000 },

{ artist: 'Justin Bieber', name: 'Purpose', sales: 554000 }

];

const popular = musicData.filter(album => album.sales > 1000000).map(album => `${album.artist} is a great performer`);

console.log(popular);

* RESET QUIZ
* TEST RUN
* SUBMIT ANSWER

**Filter & Map Quiz Solution Code**

Once you've tried your hand at solving this quiz, hover your mouse here for one possible solution.

**React is Just JavaScript Recap**

React builds on what you already know - JavaScript! You don't have to learn a special template library or a new way of doing things.

Two of the main methods that you'll be using quite a lot are:

* [.map()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map)
* [.filter()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter)

It's important that you're comfortable using these methods, so take some time to practice using them. Why not look through some of your existing code and try converting your for loops to .map() calls or see if you can remove any if statements by using .filter().