← React useState Hook | Laravel API Resource Tutorial →

Higher Order Functions In JavaScript

Without fail, you are going to need to iterate or loop over data in your programs no matter the language. In JavaScript we have several loop constructs to do this, as well as the so called higher order functions. These higher order functions allow a more concise syntax to allow you to iterate over a collection while applying the logic of your choice to each item in the collection if you like. They are sometimes a little tricky, so that is why we'll practice a bunch of them in this tutorial. We'll look at for, forEach, for of, for in, filter, map, sort, and reduce.

We Need Some Data

We can practice on a simple set of data. Here is an array of objects and an array of numbers that we'll use for practice.

Laravel File Structure

Getting Your Database Set Up For Use With Laravel

Top 12 Websites for Twitter Bootstrap

PHP MySQL CRUD Tutorial

PHP And MySQL Tutorial

Node.js Express Rest Api Tutorial

How Do Functions Work in Python?

React useState Hook

VueJs Parent Child Communication

Linux Files and Directories

Node MongoDB Native

Laravel Blade Master Template Configuration

```
// Array of objects
const companies = [
  { id: 1, name: "Square", sector: "Finance", founded:
  { id: 2, name: "Disney", sector: "Media", founded: 19
  { id: 3, name: "Ford", sector: "Auto", founded: 1903
  { id: 4, name: "Netflix", sector: "Media", founded: 19
  { id: 5, name: "Apple", sector: "Technology", founded
  { id: 6, name: "Visa", sector: "Finance", founded: 19
  { id: 7, name: "Tesla", sector: "Auto", founded: 2003
  { id: 8, name: "Microsoft", sector: "Technology", four
  { id: 9, name: "Roku", sector: "Media", founded: 200
];

// Array of numbers
const numbers = [32, 15, 20, 16, 52, 54, 11, 49, 31, 1
```

We'll also use this wrapper function so we can see the results right in the browser window rather than having to manually open the console.

```
function output(data) {
  document.write(data + "<br/>");
}
```

for

Ok! Now let's start with some basic loops so we can see how things will work a bit differently once we get to the higher order functions. First up is the basic `for` loop. We want to just loop through all of the companies and output their name.

```
for (let i = 0; i < companies.length; i++) {
  output(companies[i].name);
}
```

Square
Disney
Ford
Netflix
Apple
Visa
Tesla
Microsoft
Roku

forEach

How To Highlight New Content For Returning Visitors

How To Add Post Meta In WordPress

Twitter Bootstrap Navigation Elements

WordPress Toolbar Tutorial

New String Methods In ES6

Install Laravel on Windows

How To Set Up Form Submission In Laravel

How To Add Flash Messages

What Is NodeJS?

Using Operators in PHP

Building A Forum With Laracasts

ES6 Modules With Traceur.js

Interface Examples For Object Oriented PHP

Feature Test vs Unit Test And Adding Replies To Threads

Php Tutorials For Beginners

Send Email With Laravel

How To Install WordPress on Laravel Homestead

To do the same thing with `forEach`, we can use this style of code.

```
companies.forEach(function(company) {
  output(company.name);
});
```

Square
Disney
Ford
Netflix
Apple
Visa
Tesla
Microsoft
Roku

for of

The `for of` loop would look like this.

```
for (let company of companies) {
  output(company.name);
}
```

for in

With the `for in` loop we can iterate over all of the key value pairs of an object. With this code, we iterate over the properties of the 4th object in our companies array.

```
for (let property in companies[3]) {
  output(property + ": " + companies[3][property]);
}
```

id: 4
name: Netflix
sector: Media
founded: 1997

filter

The filter function is the first higher order function we can look at. If you write **React code**, you are going to use filter all the time! This function is used to access a subset of an array based on a condition. Say we wanted only the

Create Form Elements Using Laravel and Bootstrap
How To Use WordPress Excerpts
What Is Guzzle PHP?
Single Responsibility Principle
jQuery Event Handling
Laravel hasMany and belongsTo Tutorial
The Most Awesome PHP Script to Highlight Your Code
24 Popular Python Repositories
What Is The CSS Box Model?
ES6 Object Literal Enhancements
Open Closed Principle
Display Activity Feed In The Browser
AngularJS Component Tutorial
Laravel Eloquent ORM Tutorial
Axios Powered VueJS Form Component
CSS Position Tutorial

numbers in our numbers array that were less than or equal to 20. Before filter, we'd have to use a for loop like this.

```
let twenty = [];
for (let i = 0; i < numbers.length; i++) {
  if (numbers[i] <= 20) {
    twenty.push(numbers[i]);
  }
}
```

```
15
20
16
11
19
```

Using the `filter` function, we can accomplish the same result like this.

```
const twenty = numbers.filter(data => {
  if (data <= 20) {
    return true;
  }
});
```

```
15
20
16
11
19
```

To simplify this even further, you could use this simple one liner!

```
const twenty = numbers.filter(num => num <= 20);
```

```
15
20
16
11
19
```

Now let's work with the companies array of objects along with the filter function. Suppose we want to output only the companies in the Finance sector. No problem!


CSS Grid Tutorial


JavaScript Prototype Pattern


Introduction To C# And Visual Studio For Beginners


PHP Namespaces Tutorial


jQuery Selectors and Filters


The Most Popular Content Manipulation Methods in jQuery


How To Protect Specific Routes With Middleware


Laravel Validation


Laravel Subscription System Tutorial


ES6 Sets and Maps


Reference Types And Value Types In C#


Laravel Vue Component Example

Introduction To Laravel Controllers


Laravel Migration Generator


How To Create A Custom Menu In WordPress


WordPress Widgets Tutorial

```
const financeCompanies = companies.filter(function(cc
  if (company.sector === "Finance") {
    return true;
  }
});

financeCompanies.forEach(data => output(data.name)
```

Square
Visa

We can use the filter function to find the companies that were founded during the 1970's like so.

```
const seventiesCompanies = companies.filter(
  company => company.founded >= 1970 && company
);

seventiesCompanies.forEach(company => output(comp
```

And it looks like Apple and Microsoft were founded in the seventies.

Apple
Microsoft

Note that the name we give to the parameter for the arrow functions can be anything we want. Most times you'll use a naming convention that makes sense to you but as an example, the following code produces the same exact results.

```
const seventiesCompanies = companies.filter(
  tomato => tomato.founded >= 1970 && tomato.founc
);

seventiesCompanies.forEach(cucumber => output(cucu
```

This always confused me a little bit, so hopefully that little tip clears it up for you (and me!).

Here is another example of the filter function where we find companies that have been in business for 40 or more years. All it takes is a simple one liner. (The second line of code is just for displaying the result in the browser).

```
const fortyYears = companies.filter(company => 2019 -

fortyYears.forEach(company => output(company.name
```

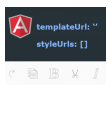


8 Steps To Success With Laravel Events


HTML Encoding With htmlspecialchars and htmlentities

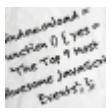

Drawing Scalable Vector Graphics With D3 JavaScript


How To Remember Form Data


Angular Templates and Styles


Angular Parent Child Communication


Laravel Cache Tutorial


JavaScript Events Tutorial


The Declarative Nature of SQL


JavaScript Module Pattern

```
Disney
Ford
Apple
Visa
Microsoft
```

Like we mentioned earlier you'll be using the **filter function with React**, so it's a good one to practice.

map

Now we can move on to the `map` function. So what does the map function do for us? The map function is used to create a new array which holds the results of calling a given function on every element in original array. So you take an array, call a function on every element, which populates a new array. Let's see a few examples of how to use the map function.

We could pluck out the names of each company for example and put that into a new array.

```
1 const newArray = companies.map(company => compar
2
3 newArray.forEach(company => output(company));
```

```
Square
Disney
Ford
Netflix
Apple
Visa
Tesla
Microsoft
Roku
```

We can use map to run a function which puts a description of the company into a new array, then output it.

```
1 const description = companies.map(
2   company =>
3     `${company.name} was founded in ${company.four
4       company.sector
5     } business`
6 );
7
8 description.forEach(d => output(d));
```

```
Square was founded in 2009 and is in the Finance business
Disney was founded in 1923 and is in the Media business
Ford was founded in 1903 and is in the Auto business
Netflix was founded in 1997 and is in the Media business
Apple was founded in 1976 and is in the Technology business
Visa was founded in 1958 and is in the Finance business
Tesla was founded in 2003 and is in the Auto business
Microsoft was founded in 1975 and is in the Technology business
Roku was founded in 2002 and is in the Media business
```

How about an example of using map to iterate over all of the numbers in our numbers array. This example will take each number in the array and set it to the power of 2. The result is stored in a new array and then we output the result to the browser.

```
1 const powerTwo = numbers.map(num => Math.pow(nu
2
3 powerTwo.forEach(num => output(num));
```

```
1024
225
400
256
2704
2916
121
2401
961
361
5625
1764
2025
4624
900
```

So those are a few examples of using the map function. To create a **list in React**, you can use the map function.

sort

Another higher order function we can take a look at is the sort function. The first example will show how to sort the companies from the oldest to the newest.

```
const sortedCompanies = companies.sort(function(c1,
  if (c1.founded > c2.founded) {
    return 1;
  } else {
    return -1;
  }
});

sortedCompanies.forEach(c => output(c.name));
```

Ford
Disney
Visa
Microsoft
Apple
Netflix
Roku
Tesla
Square

You can accomplish the same result with this much more terse code which uses the ternary operator like so.

```
const sortedCompanies = companies.sort((c1, c2) =>
  c1.founded > c2.founded ? 1 : -1
);

sortedCompanies.forEach(c => output(c.name));
```

How about sorting the numbers array from smallest to highest? That can be done with this snippet.

```
const sortInts = numbers.sort((a, b) => a - b);

sortInts.forEach(n => output(n))
```

;

```
11
15
16
19
20
30
31
32
42
45
49
52
54
68
75
```

reduce

The `reduce` function is an interesting one that takes an array with many values and reduces it down to one output value. You see this one a lot for adding up all the values in an array and then outputting that value. I always thought it kind of odd you use a reduce function to add, but alas, that is how it is.

For Example:

```
let numSum = 0;
for (let i = 0; i < numbers.length; i++) {
  numSum += numbers[i];
}

output(numSum);
```

```
559
```

You could use this more elegant solution to get the same result.

```
const numSum = numbers.reduce((total, num) => tota
```

Now let's practice using reduce on the array of company objects. This works a little differently, but the concept is the same. Let's add up the total number of years the companies have collectively been in business.

```
const totalYears = companies.reduce(function(total, co
  return total + (2019 - company.founded);
}, 0);

output(totalYears);
```

```
425
```

The terse version works equally well.

```
const totalYears = companies.reduce(
  (total, company) => total + (2019 - company.foundec
  0
);
```

An interesting aspect of the higher order functions is that you can chain them together to perform several operations in one shot. In the example here, we'll take our numbers and multiply each by three, then look at only those with a value greater than or equal to 70, and finally sort them from lowest to highest.

```
const combined = numbers
  .map(num => num * 3)
  .filter(num => num >= 70)
  .sort((a, b) => a - b);

combined.forEach(c => output(c));
```

```
90
93
96
126
135
147
156
162
204
225
```

Learn More

- Higher-Order Functions in JavaScript — SitePoint
- Higher-Order Functions :: Eloquent JavaScript
- Understanding Higher-Order Functions in JavaScript – Bits and Pieces
- Higher Order Functions – Practical introduction
- Higher Order Functions in JavaScript – Zsolt Nagy

Higher Order Functions In JavaScript Summary

In this tutorial we learned about some of the higher order functions available in JavaScript such as filter(), map(), sort(), and reduce(). By using these higher order functions you can lessen the amount of `for` loops you have to write and enable yourself to create more terse and elegant code. In addition, these types of functions are often needed for transforming or filtering data in popular frameworks like Vue, React, and Angular, so they are a good skill to have in your toolbox.

#javascript

← React useState Hook | Laravel API Resource Tutorial →