KEVIN CHISHOLM – BLOG

Web Development Articles and Tutorials

Lodash Find vs Lodash Filter

Home › JavaScript › Lodash Find vs Lodash Filter

When it comes to Lodash Find vs Lodash Filter, ask yourself: “Do I need to find ONE thing in my array, or MULTIPLE things?”

In this article, you will learn about Lodash Find vs Lodash Filter and the right time to use each. Often while coding, there is a need to find one or more values in an array. It can be quite cumbersome to find these values using if statements, and the JavaScript native array.find can also be limiting in some situations. The find and filter methods are for precisely that purpose, but before we look at which one is better, let’s quickly make sure everyone is up to speed with these terms.

Lodash

Lodash is a library similar to jQuery, Tornis, and Pixelmatch. It must be loaded before you can use it. Lodash is a popular library with many great features that help with manipulating and iterating through objects, arrays, functions, and collections.

Lodash.find()

The Lodash **.find()** method allows you to find a single item in a collection of items. The Lodash find method is similar to the JavaScript native **array.find()** except that it is more robust and lot limited to arrays.

Lodash.filter()

The Lodash **.filter()** method is very similar to the Lodash find method except that it allows you to find several items of the same type in a collection. The filter method returns an array of results.

Lodash Find vs Lodash Filter Examples

Let’s take a look at a few examples to help you see the difference. We have an array called **cars** that holds three objects. Each object is a vehicle that has three properties: name, maker, and issuv.

Example 1

```
const cars = [
  { name: 'Mustang', maker: 'Ford', isSuv: false },
  { name: 'EcoSport', maker: 'Ford', isSuv: true },
  { name: 'CR-V', maker: 'Honda',  isSuv: true }
];

const match = _.find(cars, car => { return car.name === 'Mustang'; });

console.log(match); // {name: "Mustang", maker: "Ford", isSuv: false}
```

lodash-find-vs-lodash-filter-example-1.js hosted with ❤ by GitHub view raw

In Example 1, we want to find an object named Mustang. There are only a few names, and each name is different, so the Lodash find method is a perfect choice for this example. The find method returns the first Mustang that it sees, and console.log prints it to the screen.

Example 2

```
const cars = [
    { name: 'Mustang', maker: 'Ford', isSuv: false },
    { name: 'EcoSport', maker: 'Ford', isSuv: true },
    { name: 'CR-V', maker: 'Honda',  isSuv: true }
];

const match = _.find(cars, car => { return car.isSuv; });

console.log(match); // {name: "EcoSport", maker: "Ford", isSuv: true}
```

lodash-find-vs-lodash-filter-example-2.js hosted with ❤ by GitHub view raw

In Example 2, we demonstrate a problem with the Lodash find method. In this case, we want to know which objects are SUVs. If we use the Lodash find method this time, it will return the object named EcoSport, and console.log will print it to the screen. It will not see that the CR-V is also an SUV because it stops when it finds a match. The same problem will occur if you inquire about which objects are made by Ford.

Example 3

```
const cars = [
    { name: 'Mustang', maker: 'Ford', isSuv: false },
    { name: 'EcoSport', maker: 'Ford', isSuv: true },
    { name: 'CR-V', maker: 'Honda',  isSuv: true }
];

const match = _.filter(cars, car => { return car.isSuv; });

console.log(match);
// 0: {name: "EcoSport", maker: "Ford", isSuv: true}
// 1: {name: "CR-V", maker: "Honda", isSuv: true}
```

lodash-find-vs-lodash-filter-example-3.js hosted with ❤ by GitHub view raw

In Example 3, we demonstrate that to find multiple objects, we will need to use the Lodash filter method. This time we will try again to see which objects are SUVs, but this time we will use the Lodash filter method. The filter method will return an array containing every object that is an SUV. In this case, the result will be the EcoSport and the CR-V.

This method will also work correctly to find which vehicles are made by Ford.

Conclusion

That’s all there really is to know about these two methods. Lodash Find vs Lodash Filter boils down to whether or not you need more than one result. Both of these methods can be potent tools at your disposal and open up options that might take hundreds of lines of code to write yourself. However, like any library, Lodash can slow down your load times, and is overkill for small projects like these examples. Whenever you can, use regular JavaScript, but keep these options open.

If you have enjoyed reading over these examples and have found them to be a new way to improve your code, please share this Lodash Find vs Lodash Filter post on Facebook and Twitter.