Tip 6: Check Existence in an Array with Includes()

In this tip, you'll learn how to find out if a value exists in an array without checking position.

We'll cover the following

It's easy to get so caught up in the big exciting changes in a language (such as the spread operator, which you'll see in a moment) that you miss the small changes that simplify common tasks.

Testing existence

Arrays now have an easy improvement to handle a common problem: **testing existence**. Testing existence is an important action, and it's crucial in everything from ternaries (Tip 18, Check Data Quickly with the Ternary Operator), to short circuiting (Tip 19, Maximize Efficiency with Short Circuiting), to most conditionals in general.

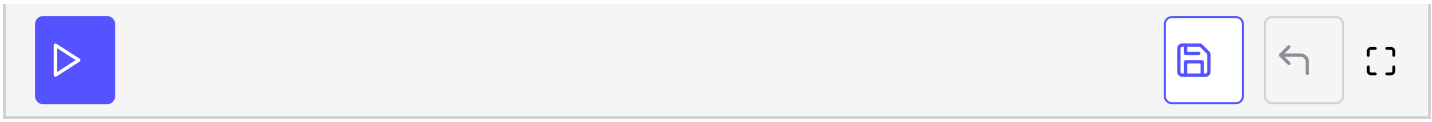
Using `indexOf`

Testing existence with JavaScript arrays has always been a little clunky. For example, if you want to see if an array contains a certain string, you check to see if the string has a position (position being another feature of an iterable). If the position exists, you'll get the *index*. If not, you get `-1`. The problem is that the index can be 0, which evaluates to false (also known as being falsy). This means the existence can be true, but the check can evaluate to false.

```
const sections = ['shipping'];

function displayShipping(sections) {
    if (sections.indexOf('shipping')) {
        return true;
    }
    return false;
}
```

```
console.log(displayShipping(sections));
```

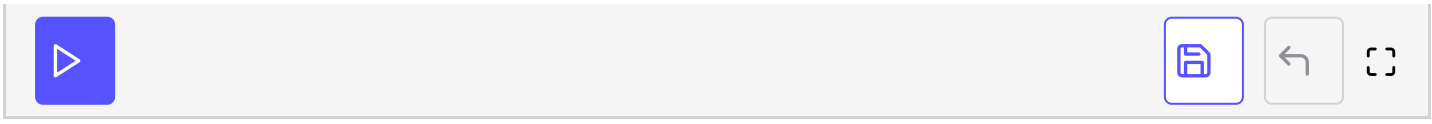


Because of this unfortunate situation, a position at `0` being falsy, you have to compare the index against a number and not just test that it's truthy. It's not a big problem, but it's just extra code to remember. Jump ahead to Tip 17, Shorten Conditionals with Falsy Values, for more on falsy values.

```
const sections = ['contact', 'shipping'];

function displayShipping(sections) {
    return sections.indexOf('shipping') > -1;
}

console.log(displayShipping(sections));
```



Using `includes`

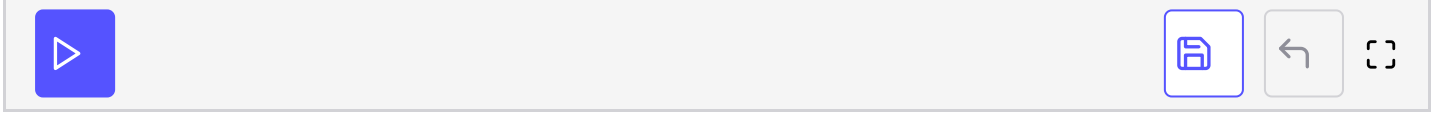
Fortunately, another feature coming up in ES2017 will eliminate that boilerplate comparison. The new array method, called includes(), will check to see if a value exists in an array and return a Boolean of `true` or `false`.

You can rewrite the preceding code with a simple check.

```
const sections = ['contact', 'shipping'];

function displayShipping(sections) {
    return sections.includes('shipping');
}

console.log(displayShipping(sections));
```



This may seem like a trivial change, but after writing `-1` over and over in a codebase, or even worse, forgetting and getting false negatives on a zero-indexed value, it's a welcome change.

Now that you've seen how integral arrays are to JavaScript, you'll dive into them a little more as we explore some of the new features that make them even more

exciting and powerful. It's best to get comfortable with arrays because they're everywhere in JavaScript. And even if you aren't using them directly, don't be surprised if a lot of what you learn about arrays begins to show up in other collections.

In the next tip, you'll learn how to use the most interesting and powerful new technique for working with arrays: the *spread operator*.