Using forEach() will not be useful if you want to permanently modify the original array. forEach() always returns undefined. However, creating a new array from an existing array is simple with the powerful map() method.

With the map() [**method**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map), you can take an array, perform some operation on each element of the array, and return a new array.

**var** donuts = ["jelly donut", "chocolate donut", "glazed donut"];

**var** improvedDonuts = donuts.map(**function**(donut) {

donut += " hole";

donut = donut.toUpperCase();

**return** donut;

});

**donuts array:** ["jelly donut", "chocolate donut", "glazed donut"]

**improvedDonuts array:** ["JELLY DONUT HOLE", "CHOCOLATE DONUT HOLE", "GLAZED DONUT HOLE"]

Oh man, did you just see that? The map() method accepts one argument, a function that will be used to manipulate each element in the array. In the above example, we used a function expression to pass that function into map(). This function is taking in one argument, donut which corresponds to each element in the donuts array. You no longer need to iterate over the indices anymore. map() does all that work for you.

NEXT