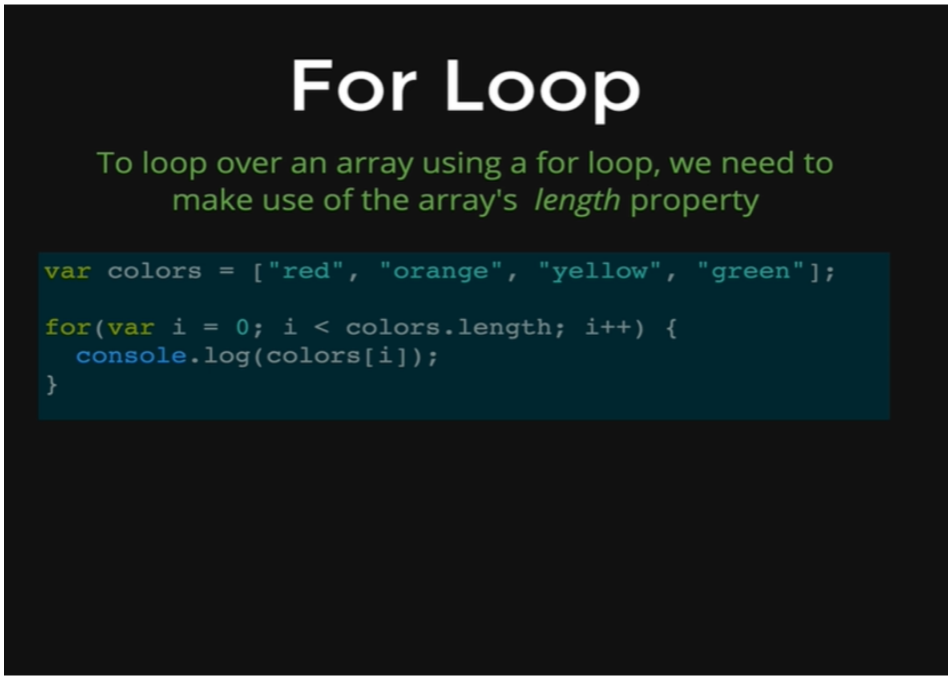
Objectives

- *Use a for loop to iterate over an array*
- *Use forEach to iterate over an array*
- *Compare and contrast for loops and forEach*

In this lesson we will continue to learn about Arrays. Here we are going to focus on Array Iteration. Until this point we have not learned how to access every item of an array, or do some code to every item in an array, and that's what array iteration refers to, iterating over a list, iterating over an array, basically looping through that array and doing something to each item or with each item.

Let's talk about some real-world examples of array iteration. A common example is *comment* and *posts.* A blog post like the ones in Reddit – a Reddit post has a bunch of comments, well those comments are stored in an array, and in order to display all of those comments, some code would loop or iterate through that array with 10,000 comments potentially and for each one it makes some HTML contents, it makes some *li*s, or it makes a paragraph (<p>), whatever the comments are, that HTML is generated in a loop, when we iterate over an array. We will be doing something very similar with comments as well.

Array Iteration is really important, it is probably the most common thing that we do with Arrays, we have a list of data, usually we want to do something in that list, otherwise if you only want to work with the first thing always or the seventh thing always, maybe we should not be using an array, maybe we just store that item on its own.



For Loop

To loop over an array using a for loop, we need to make use of the array's *length* property

```
var colors = ["red", "orange", "yellow", "green"];

for(var i = 0; i < colors.length; i++) {
  console.log(colors[i]);
}
```

Here we have an example of a *for loop* to loop through an array. We have an array of colors, that has four items, red, orange, yellow and green and if we want to print out each one and we do not know about array iteration, then what we need to do is as below.

var colors = [“red”, “orange”, “yellow”, “green”];

We have declared our array in a variable, now we will start printing each item one by one.

console.log(colors[0])

- ➔ *red*

console.log(colors[1])

- ➔ *orange*

console.log(colors[2])

- ➔ *yellow*

console.log(colors[3])

- ➔ *green*

As you can see that it is obnoxious and that it is as bad as it could be if we had 10,000 colors in this array and 10,000 comments in a Reddit post. That obviously would not work for us. Not to mention that it is also not DRY code, we are repeating ourselves all the time.

That is why we use a loop to help automate this process, what we were doing by console.log is the same operation and the only thing that changes is the number inside the square brackets, and each time we are just adding 1, we are starting at the beginning of our array, and we are going all the way to our end.

We can do all these with a *for loop*, as we can see in the picture, our *for loop* will start with i equal to 0, because that is always the first index in the array and then we add 1 to i each time to the loop and we keep going while i is less than the length of the colors’ array, so remember the length of the array is four, and while i is less than four, it should never be equal to four as the length of our array is four, and the index of our last item is 3, which is one less than the length of our array. That is why *colors[4]* is not defined.

colors[4]

- ➔ *undefined*

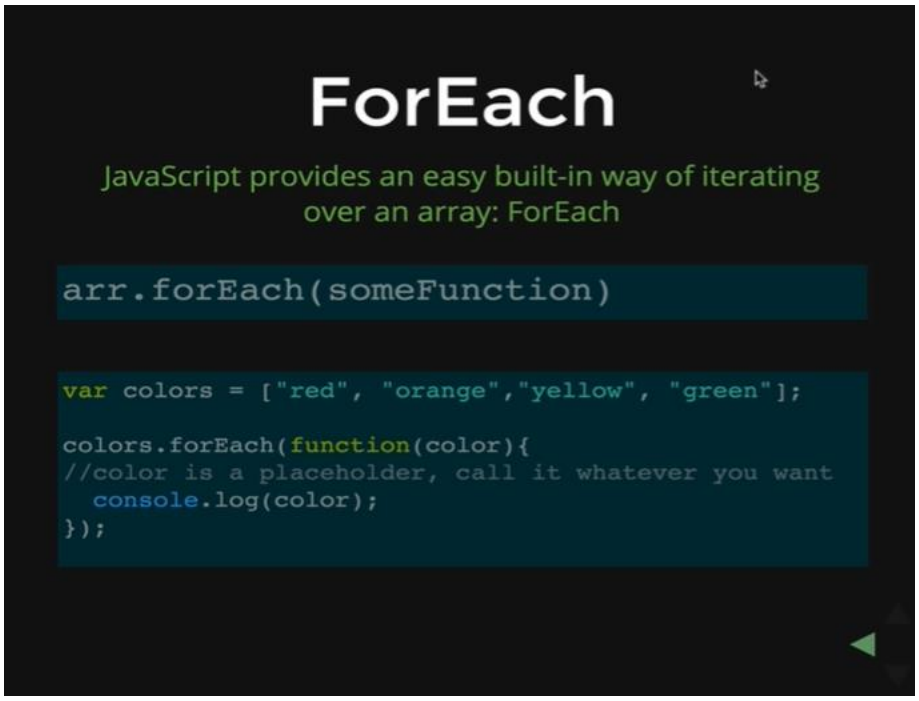
```
for(var i = 0; i < colors.length; i++) {
	console.log(colors[i]);
}
```

- ➔ *red*
- ➔ *orange*
- ➔ *yellow*
- ➔ *green*

To make comments in an application we can write something like

```
for(var i = 0; i < comments.length; i++) {
	makeCommentHTML(comment[i]);
}
```

And this would be responsible for making all the HTML for one comment, and now we are doing it to every item in the comments' array. That's how we use a *for loop*, we can also go backwards, so we start at the end of the array and we keep going till we hit 0.



ForEach

JavaScript provides an easy built-in way of iterating over an array: ForEach

```
arr.forEach(someFunction)
```

```
var colors = ["red", "orange","yellow", "green"];

colors.forEach(function(color){
//color is a placeholder, call it whatever you want
  console.log(color);
});
```

There is whole other way of iterating through an array called a *forEach*, and in our opinion *forEach* is much nicer to use, its simpler, its shorter and its much common now a days. *forEach* has not always been a part of JavaScript, so it is relatively new compared to using a *for loop*, a plain old *for loop* to loop through an array. It came out around 2009 or so. There is a little bit of hurdle to get over it at the very beginning, as you can see on the picture that we are passing in a function into another function. Once we get passed to that, once we learn the syntax and get used to it, and we know when we need to have brackets and parentheses, there is kind of a little bit of baggage that we have to get through at the beginning, but once we are comfortable with it, it is so much easier than using a *for loop*. Its much faster and we will see it all over the place.

It's a method called *forEach* that is defined in every single array, its part of something called the array prototype, which is where all those methods like *push, pop, shift and unshift*, its where they all live. What it does is that it takes a function as an argument. So, we first write the name of the array and then we give a dot and write *forEach* and then we pass in a function inside the parentheses as an argument.

arr.forEach(someFunction)

It does not always look like that as in the picture. Normally, what it looks like is an anonymous function like below.

```
colors.forEach(function(color){
        //color is a placeholder, call it whatever you want
        console.log(color)
        });
```

The function that we see, which is holding the *color* argument is called for every single element in the array. Let's demonstrate this through an example.

```
var colors = [“red”, “orange”, “yellow”, “green’];
colors.forEach(function(){
        console.log(“INSIDE THE FOREACH”);
        }
```

Output:

```
INSIDE THE FOREACH
INSIDE THE FOREACH
INSIDE THE FOREACH
INSIDE THE FOREACH
```

This function is called for all the items that we have inside our array, as we have stored only four items in our array, that is why *INSIDE THE FOREACH* is printed for each four items.

Though it is not useful enough to run some code *x* number of times for an array, we usually want to use the data in the array somehow, whether its making common HTML, whether its saving something to the database, whether its adding to the score of each item, but we usually interact or manipulate that data in someway rather than just arbitrarily alerting or console.logging.

The way we get that data is by having our function, inside the *forEach*, take an argument and use that argument in the iteration of the array. Let's demonstrate this case using an array where we will use an argument to print all the colors in our array with “INSIDE THE FOREACH” that we printed in our last code.

```
colors.forEach(function(x){
        console.log(“INSIDE THE FOREACH ” + x);
        });
```

- ➔ *INSIDE THE FOREACH red*
- ➔ *INSIDE THE FOREACH orange*
- ➔ *INSIDE THE FOREACH yellow*
- ➔ *INSIDE THE FOREACH green*

What is happening here is that the variable *x* that we passed as an argument is holding the value of each item in that array, as the *forEach* loops through it is calling the function in each item of the array, and that is why at every item, the function is printing INSIDE THE FOREACH and after that the name of the item or the color it is on.

Let's demonstrate this case with a syntax.

```
function printColor(color) {

        console.log("*********************");

        console.log(color);

        console.log("*********************");

        }
```

We have defined a simple function; it takes an argument called color and after we press enter nothing happens because we only defined the function. Now, after we call it lets see what happens.

```
function printColor("purple")
```

```
*************************

purple

*************************
```

We get stars and then purple and then stars again. Now we can run *forEach* to run for every item in our array. One thing we need to note is that we cannot use parentheses after our function name because then JavaScript would just execute the function instead of letting *forEach* to iterate through the array. That is why we cannot use parentheses after the function name when we are using *forEach.*

colors.forEach(printColor);

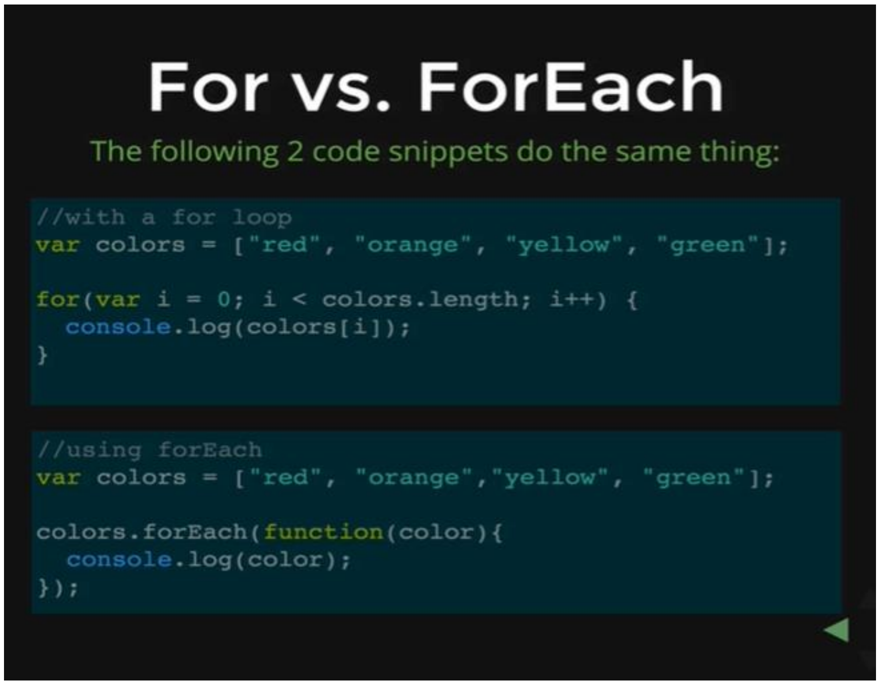
Output

```
***************************
red
***************************
***************************
orange
***************************
***************************
yellow
***************************
***************************
green
***************************
```

What happened behind the scenes is that *forEach* is taking the *printColor()* function and its iterating through the colors' array and while being at each item, it takes the item's index number and places it in the argument as *printColor(colors[0]),* and then the function prints red, as red is our first item in the array and then it moves to the second item and as *printColor(colours[1]),* it prints orange and so on.

We just tell it what to call and it will call it on the items. *forEach* is really nice because we do not have to work with *i,* we do not have to write a syntax for a for loop and also we do not need to use *i* inside the function body, all we have to do is just pass the argument with the function that we are passing in the *forEach* and it will call the function on every items as it iterates through.

We can use both of them *for loop* or, *forEach,* but *forEach* is newer, its more popular.



For vs. ForEach

The following 2 code snippets do the same thing:

```
//with a for loop
var colors = ["red", "orange", "yellow", "green"];

for(var i = 0; i < colors.length; i++) {
  console.log(colors[i]);
}
```

```
//using forEach
var colors = ["red", "orange","yellow", "green"];

colors.forEach(function(color){
  console.log(color);
});
```

Here is a comparison of two ways of looping through, but there are more than two ways as we can also use a *while* loop to iterate through the array.

```
var count = 0;
while(count < colors.length) {
        console.log(colors[count]);
        count++;
}
```

- ➔ *red*
- ➔ *orange*
- ➔ *yellow*
- ➔ *green*

It is a little bit more syntax, as we need to initialize the count variable before we define our while loop and then we need to increment the count as we loop through our array. Then we end up with red, orange, yellow and green.

We can use a while loop, but its rare, we do not see many people using it, it is almost always a *for loop*, or a *forEach*, increasingly its pretty much always a *forEach*. That's why it is always good to remember, how we write a *for loop* or to use a *forEach*. The key difference between a *for loop* and a *forEach* is what the code in the picture above shows us, the key difference is in a *for loop*, we are dealing with a number, so we are initializing variable *i* to the number to go from 0 until the end of the array, and then we are using that number to access the array. In case of *forEach*, that number is abstracted away from us, all that we are doing is we are using a name that we created as an argument, a temporary placeholder, which can be a color, or an item or a thing, comments, posts, friends, whatever it is, and we use that inside of a function and most often we will see an anonymous function inside the *forEach* unless there is a function that we want to use later on, or we need to use it at some other part of the code then we might define it, or we might give it a name outside the *forEach.*