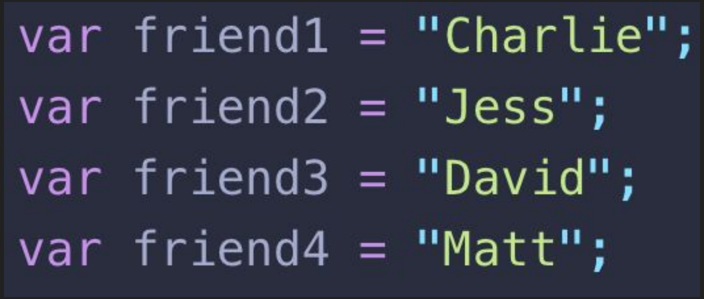
JavaScript

Arrays

Arrays

- Suppose I wanted to model a group of friends:

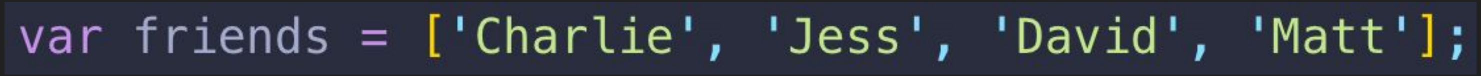


```
var friend1 = "Charlie";
var friend2 = "Jess";
var friend3 = "David";
var friend4 = "Matt";
```

- This is a lot of code, and it doesn’t let us group the friends together.

Arrays Cont.

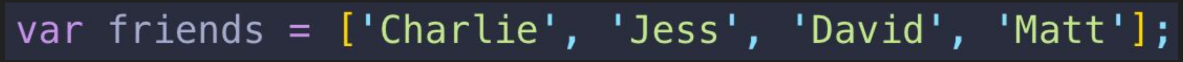
- This is a perfect example use case for an ARRAY.



```
var friends = ['Charlie', 'Jess', 'David', 'Matt'];
```

Array’s cont.

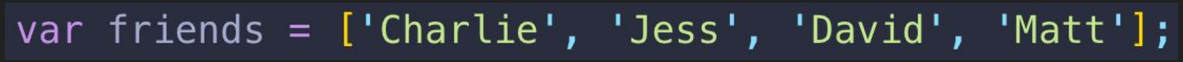
- Arrays let us group data together in lists.



```
var friends = ['Charlie', 'Jess', 'David', 'Matt'];
```

- Arrays are indexed starting at 0. Every slot has a corresponding number.

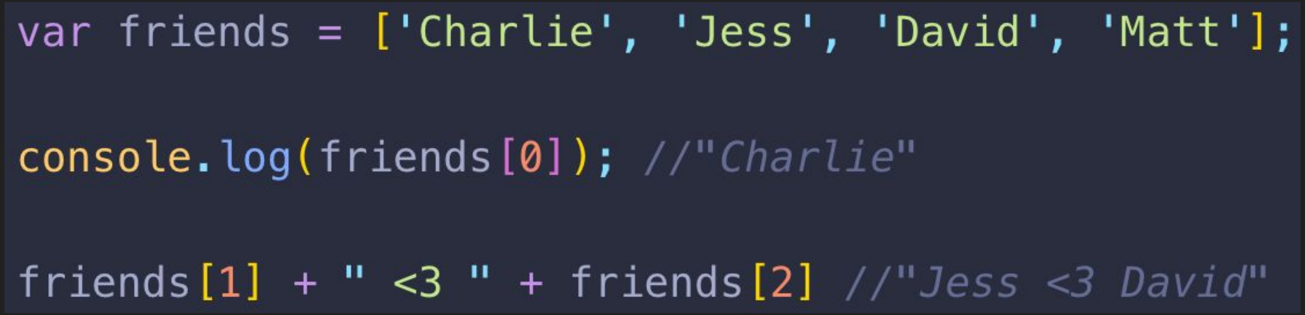
```
              0           1          2          3
```



```
var friends = ['Charlie', 'Jess', 'David', 'Matt'];
```

Arrays cont.

- We can use those indices to retrieve data.



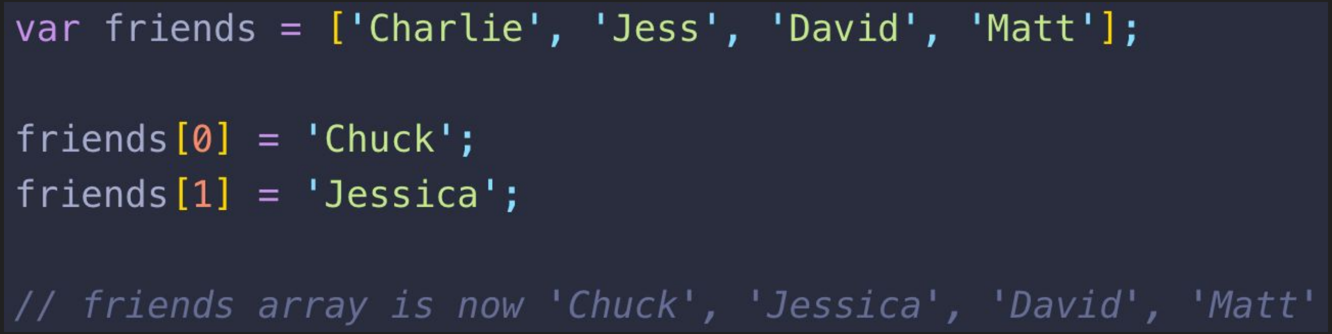
```
var friends = ['Charlie', 'Jess', 'David', 'Matt'];

console.log(friends[0]); //"Charlie"

friends[1] + " <3 " + friends[2] //"Jess <3 David"
```

Arrays cont.

- We can also update arrays:



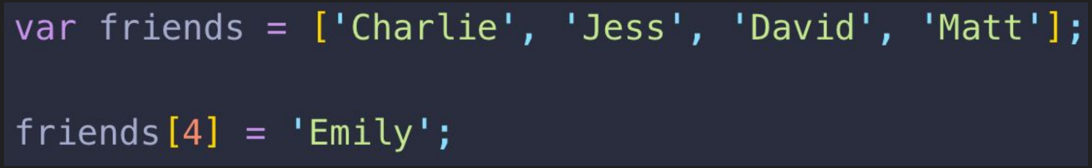
```
var friends = ['Charlie', 'Jess', 'David', 'Matt'];

friends[0] = 'Chuck';
friends[1] = 'Jessica';

// friends array is now 'Chuck', 'Jessica', 'David', 'Matt'
```

Arrays cont.

- We can also add new data.

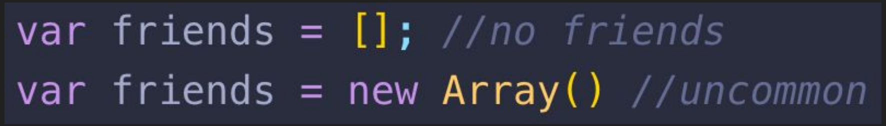


```
var friends = ['Charlie', 'Jess', 'David', 'Matt'];

friends[4] = 'Emily';
```

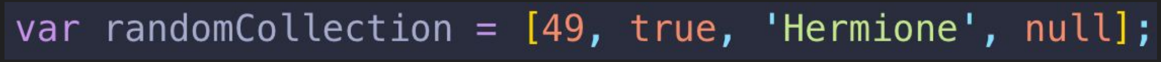
Arrays cont.

- We can initialize an empty array two ways:



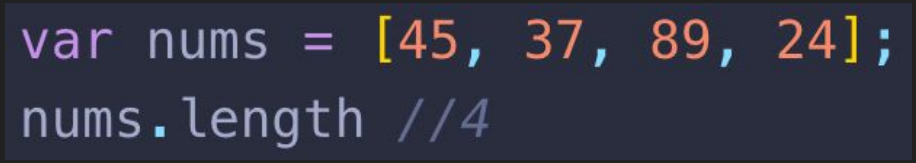
```
var friends = []; //no friends
var friends = new Array() //uncommon
```

- Arrays can hold any type of data



```
var randomCollection = [49, true, 'Hermione', null];
```

- Arrays have a length property.



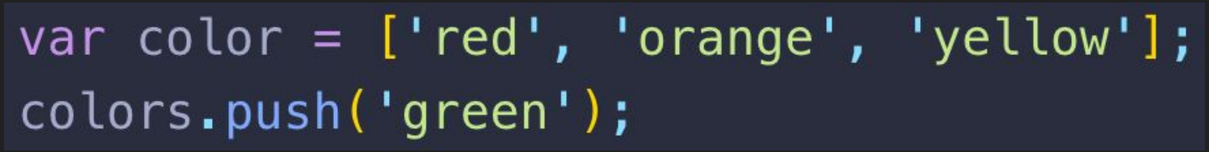
```
var nums = [45, 37, 89, 24];
nums.length //4
```

Array Methods

- Arrays come with a few built-in methods that make our life easier. We’re going to cover:
    - push/pop
    - shift/unshift
    - indexOf
    - slice

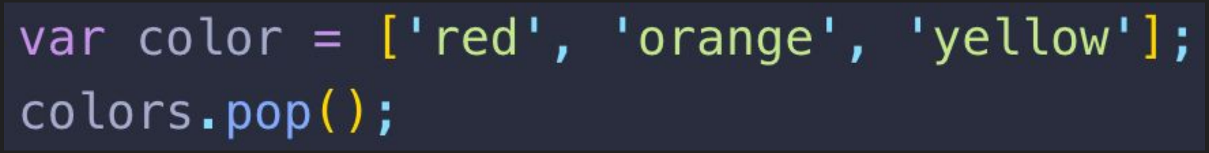
Push and Pop

- Use push to add to the end of an array:



```
var color = ['red', 'orange', 'yellow'];
colors.push('green');
```

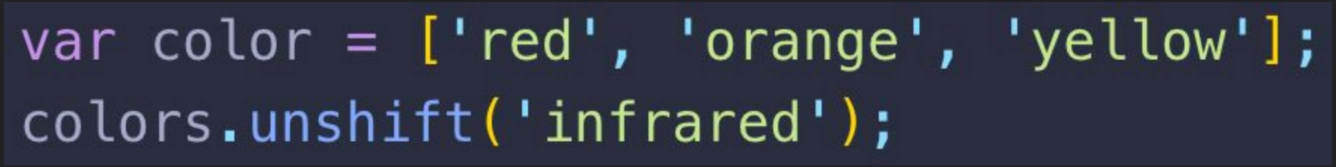
- Use pop to remove the last item in an array



```
var color = ['red', 'orange', 'yellow'];
colors.pop();
```

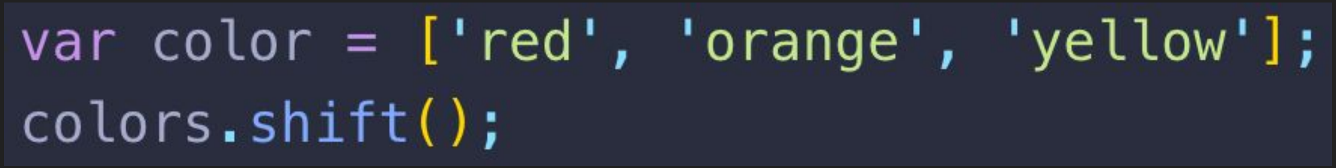
Shift and Unshift

- Use unshift to add to the front of an array:



```
var color = ['red', 'orange', 'yellow'];
colors.unshift('infrared');
```

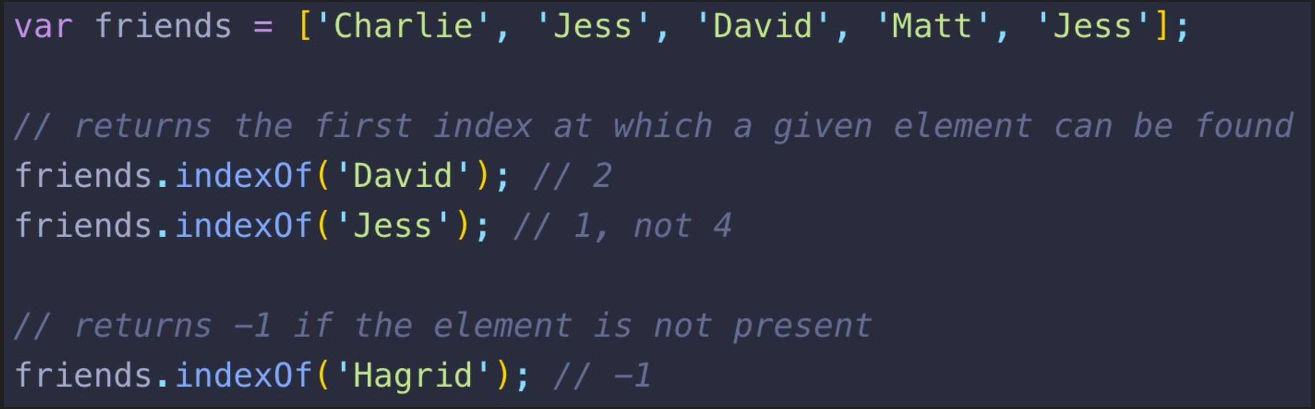
- Use shift to remove the first item in an array.



```
var color = ['red', 'orange', 'yellow'];
colors.shift();
```

IndexOf

- Use indexOf() to find the index of an item in an array



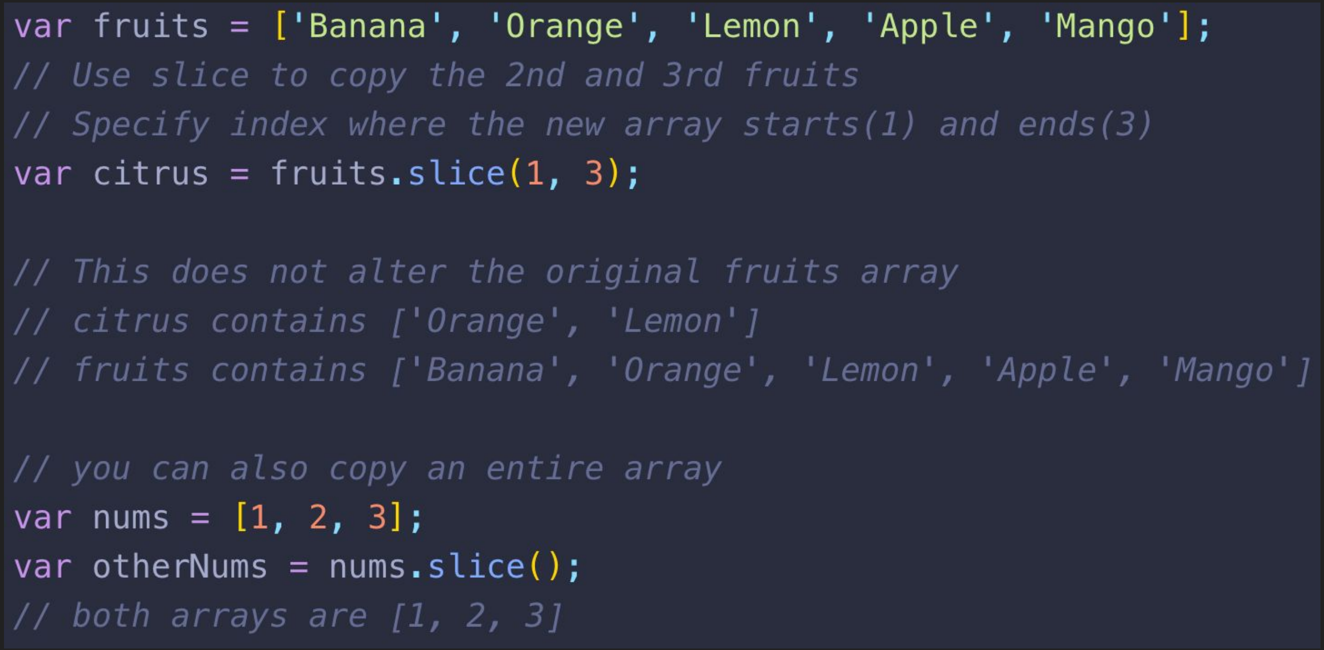
```
var friends = ['Charlie', 'Jess', 'David', 'Matt', 'Jess'];

// returns the first index at which a given element can be found
friends.indexOf('David'); // 2
friends.indexOf('Jess'); // 1, not 4

// returns -1 if the element is not present
friends.indexOf('Hagrid'); // -1
```

Slice

- Use slice() to copy parts of an array



```
var fruits = ['Banana', 'Orange', 'Lemon', 'Apple', 'Mango'];
// Use slice to copy the 2nd and 3rd fruits
// Specify index where the new array starts(1) and ends(3)
var citrus = fruits.slice(1, 3);

// This does not alter the original fruits array
// citrus contains ['Orange', 'Lemon']
// fruits contains ['Banana', 'Orange', 'Lemon', 'Apple', 'Mango']

// you can also copy an entire array
var nums = [1, 2, 3];
var otherNums = nums.slice();
// both arrays are [1, 2, 3]
```

Assignment 9.1: Todo List part 1

- Prompt the user for their input on what they would like to do.
- If the user says “new”
  - Prompt them to add a new todo.
- If the user says “list”
  - Print to the console their list.
- If the user says “quit”
  - Print to the console “you have quit the app.”
- The app must repeat until they say “quit”.

Click here for an example.

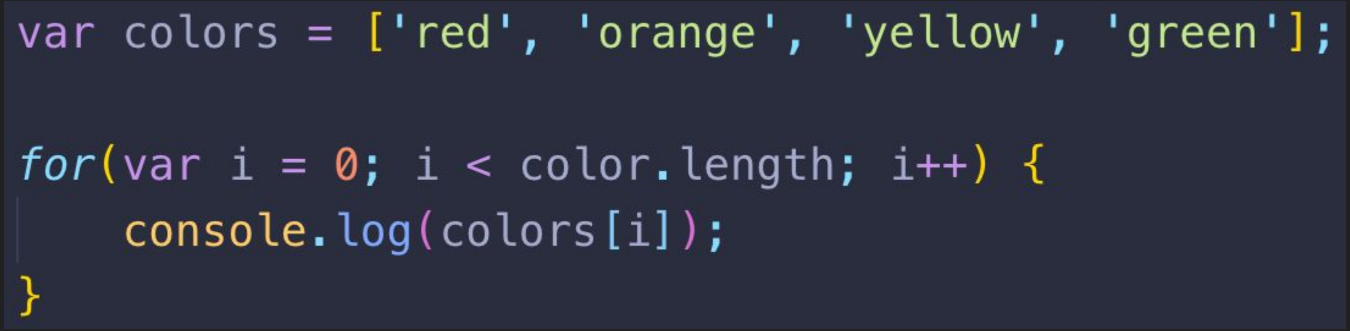
NOTE: When you go to the example, type “quit” first then open the JS console. Refresh the page then use the other commands.

Array Iteration

- You can loop through an Array by using either a for loop or a forEach loop.

Array Iteration: For loops

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

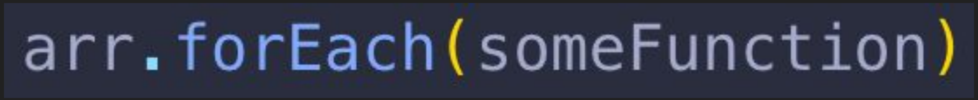


```
var colors = ['red', 'orange', 'yellow', 'green'];

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

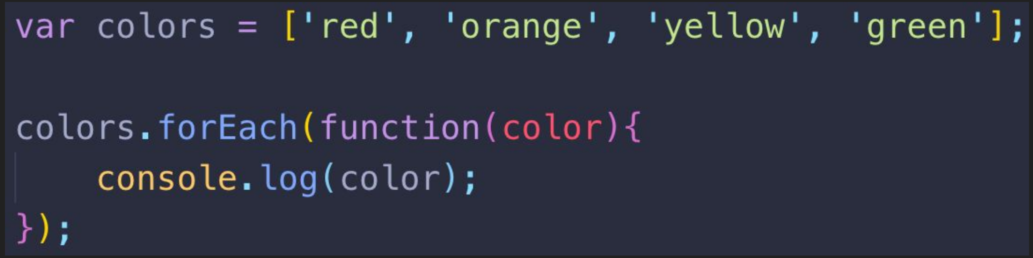
Array Iteration: ForEach loops

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



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

- Color is a placeholder, call it whatever you want.

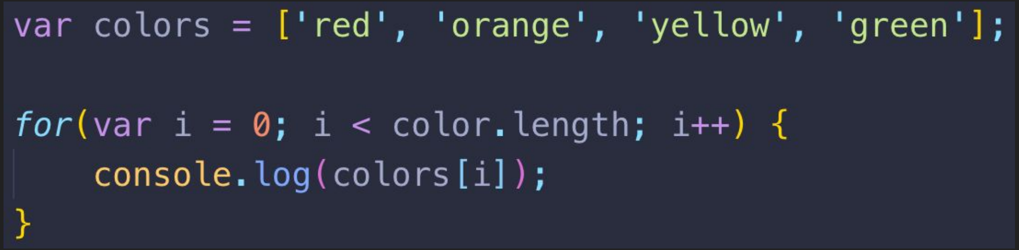


```
var colors = ['red', 'orange', 'yellow', 'green'];

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

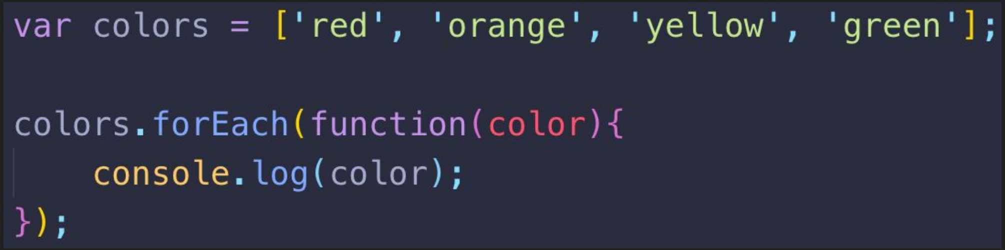
Array Iteration: For vs. ForEach

The following 2 code snippets do the same thing:



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

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

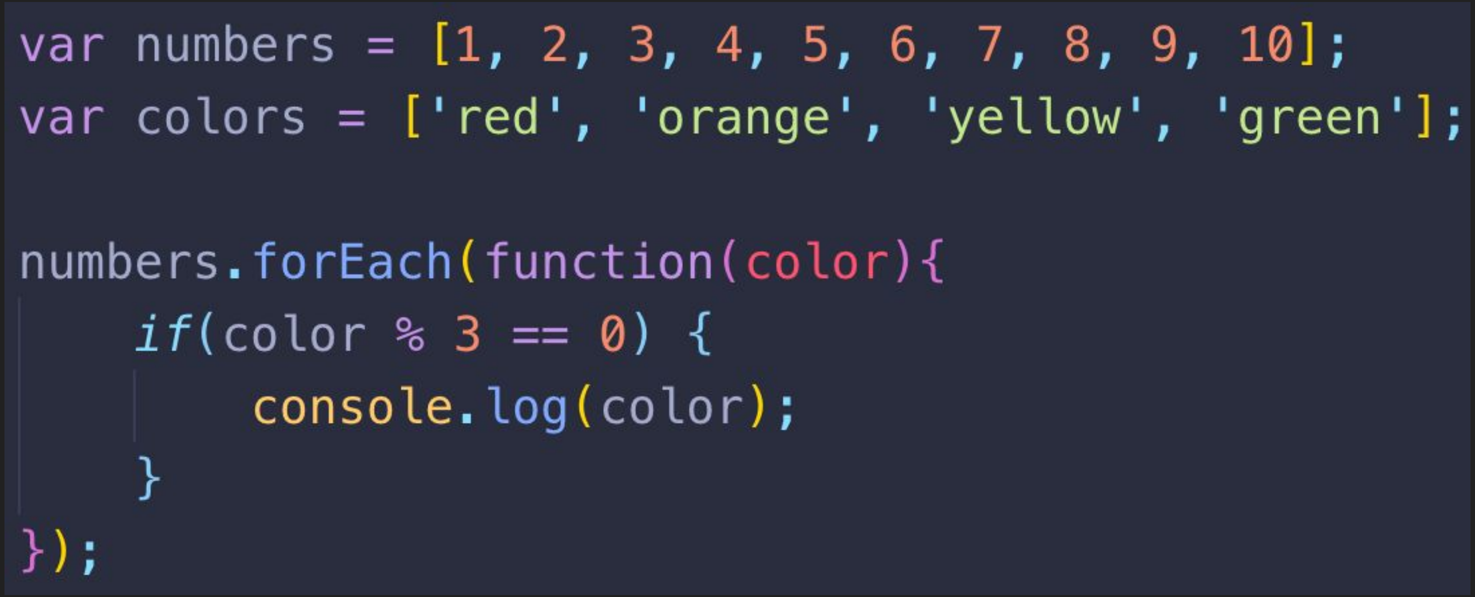


```
var colors = ['red', 'orange', 'yellow', 'green'];

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

Array Iteration Exercise

- What does the following code print out?



```
var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
var colors = ['red', 'orange', 'yellow', 'green'];

numbers.forEach(function(color){
    if(color % 3 == 0) {
        console.log(color);
    }
});
```

Assignment 9.2: Todo List part 2

- List the Array items on their own line along with their index (number) position within the console when a user says “list”.
  - Use a .forEach loop. Also, pass a second argument through the function to get the index position.
- Add a “delete” command.
  - If a user says “delete”, prompt them “Which index would you like to delete?”.
  - Based on the number they input, delete the associated Array item.
- Add a confirmation message that prints to the console when they add or delete an item.
  - When they add a new item it should print “Added new todo”
  - When they delete an item it should print “Deleted todo”

Click here for example.

Assignment 9.3: Array Problem Set

- Write a function *printReverse()* that takes an array as an argument and prints out the elements in the array in reverse order (don’t actually reverse the array itself)
- Write a function *isUniform()* which takes an array as an argument and returns true if all elements in the array are identical
- Write a function *sumArray()* that accepts an array of numbers and returns the sum of all numbers in the array.
- Write a function *max()* that accepts an array of numbers and returns the maximum number in the array.