
Globant
we are ready

UI Engineering Studio. Day 6

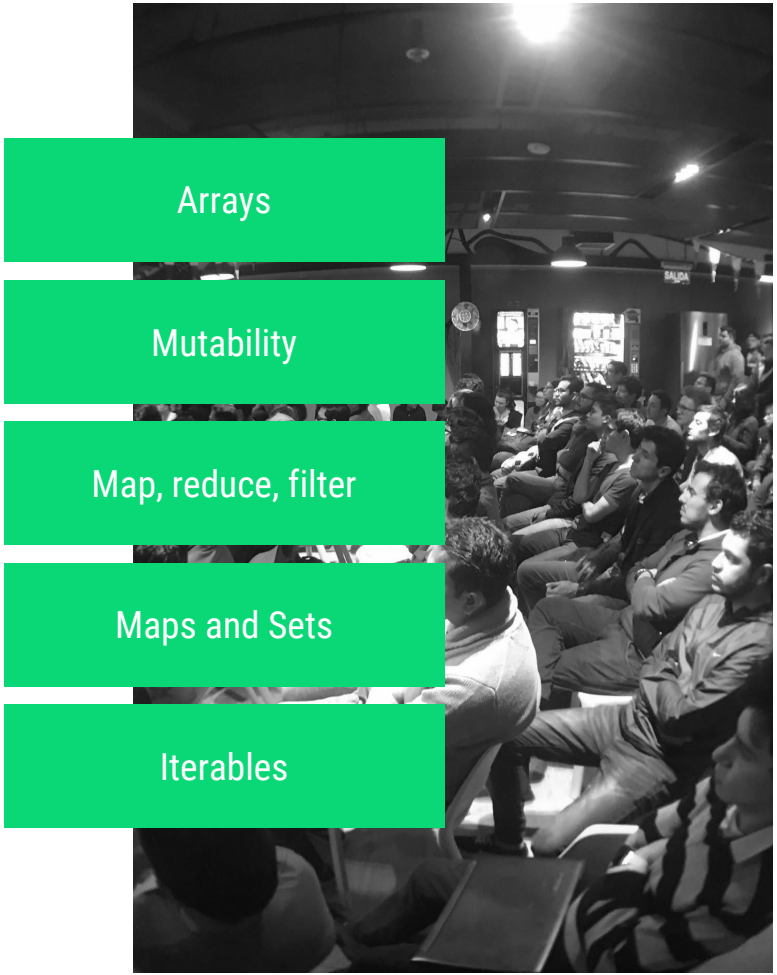
Bootcamp: JS Collections

Globant

Collections

A **collection** (...) is a grouping of some variable number of data items (possibly zero) that have some shared significance to the problem being solved and need to be operated upon together in some controlled fashion.

– Wikipedia



UI Boot Camp: JS Collections

Arrays

an **array**, is a data structure consisting of a collection of *elements* (values or variables), each identified by at least one *array index* or *key*

– Wikipedia

In Javascript, arrays can contain any number of elements from any type (even items of different types). The size of the array is not fixed.

```
var numbers = [1, 2, 3, 4, 5];
var names = ['Andrés', 'Gabriel',
'Johnny'];
var strageList = [1, true, 5.6546,
'Cristian', null];
```

UI Boot Camp: JS Collections

Array operations

Array elements are accessible by their **index (starting at zero)**. Array also have a length property, which returns the **total** number of elements contained in the array.

Assume that:

```
var numbers = [1, 2, 3, 4, 5];
```

Access an element by index

```
numbers[3]; // 4
```

Getting an array's total elements

```
numbers.length;
```

Access an element (right to left)

```
numbers[numbers.length - 2];
```

UI Boot Camp: JS Collections

Array Methods

Find the index of an item

```
numbers.indexOf(2); // 3
```

Assert if an item is inside an array

```
numbers.includes(7); // false
```

concatenate arrays

```
[1].concat([2]); // [1, 2]
```

UI Boot Camp: JS Collections

Modifying Arrays

Append an element

```
numbers.push(6); // returns 6
// numbers = [1, 2, 3, 4, 5, 6]
```

Remove last element

```
numbers.pop(); // returns 6
// numbers = [0, 1, 2, 3, 4, 5]
```

Prepend an element

```
numbers.unshift(0); // returns 0
// numbers = [0, 1, 2, 3, 4, 5, 6]
```

Remove first element

```
numbers.shift(); // returns 0
// numbers = [1, 2, 3, 4, 5]
```

UI Boot Camp: JS Collections

Mutability

In JavaScript, an Array is a mutable data structure, i.e: it can be modified in after it was created. Many array methods modify the target array in place (for example: removing an element or adding a new one).

However, Js arrays can behave like an immutable data structure by using the appropriate procedures.

Advise: avoid mutability!

UI Boot Camp: JS Collections

Modifying Arrays (immutable)

Append an element

```
numbers.concat([6]);
// returns [1, 2, 3, 4, 5, 6]
```

Prepend an element

```
[0].concat(numbers);
// returns [0, 1, 2, 3, 4, 5, 6]
```

Remove last element

```
numbers.slice(0, -1);
// returns [0, 1, 2, 3, 4, 5]
```

Remove first element

```
numbers.slice(0);
// returns [1, 2, 3, 4, 5]
```

UI Boot Camp: JS Collections

Traversing Arrays

There are 3 basic ways* for traversing an array:

1. Using a regular **for loop**.
2. Using the **for..of** loop for iterables.
3. Using the **forEach** method.

*You can still traverse an array using a while loop, but the three ways listed above are the most common

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

```
for (let number of numbers) {
 console.log(number);
}
```

```
numbers.forEach(number => console.log(number));
```

UI Boot Camp: JS Collections

Map, Filter, Reduce

Map: creates a new array with the results of calling a provided function on every element in the calling array.

Filter: creates a new array with all elements that pass the test implemented by the provided function.

Reduce: applies a function against an accumulator and each element in the array (from left to right) to reduce it to a single value.

-MDN

```
var numbers = [1, 2, 3, 4, 5];

var squaredNumbers =
 numbers.map(number => number * number);

var onlyBig =
 numbers.filter(number => number > 2);

var total =
 numbers.reduce((sum, number) => sum +
number, 0);
```

UI Boot Camp: JS Collections

Maps and Sets

Sets

```
var mySet = new Set();
mySet.add(1); // Set [ 1 ]
mySet.add(5); // Set [ 1, 5 ]
mySet.add(5); // Set [ 1, 5 ]
mySet.has(1); // true
```

Sets are collections of **unique values** of any type. If you try to insert a duplicated item in a set, it will ignore the entry.

Maps

```
var map = new Map();
map.set('name', 'Gabriel');
map.set('last name', 'Martinez');
map.set('age', 47);
map.get('last name'); // "Martinez"
```

Maps hold **key-value pairs**. Any value (both objects and primitive values) may be used as either a key or a value.

UI Boot Camp: HTML

Homework: Calculator

https://www.sitepoint.com/dom-manipulation-vanilla-javascript-no-jquery/


Globant
we are ready