JavaScript Arrays and Collections

Zawartość

[Method to use when working with arrays 2](#_Toc68454410)

[Let’s start 2](#_Toc68454411)

[Spread operator 3](#_Toc68454412)

[Find and FindIndex 3](#_Toc68454413)

[Fill 4](#_Toc68454414)

[Iteration 4](#_Toc68454415)

[Data Collection 5](#_Toc68454416)

[Intro 5](#_Toc68454417)

[Set 6](#_Toc68454418)

[Manipulate data in sets 7](#_Toc68454419)

[Weak Set 8](#_Toc68454420)

[Map 10](#_Toc68454421)

[Difference between Map and Object 12](#_Toc68454422)

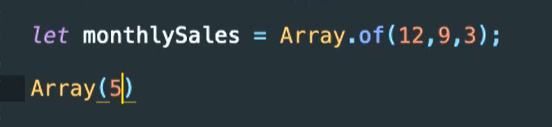
[Weak Map 12](#_Toc68454423)

[Typed Array 13](#_Toc68454424)

[Create Buffer 13](#_Toc68454425)

# Method to use when working with arrays

## Let’s start

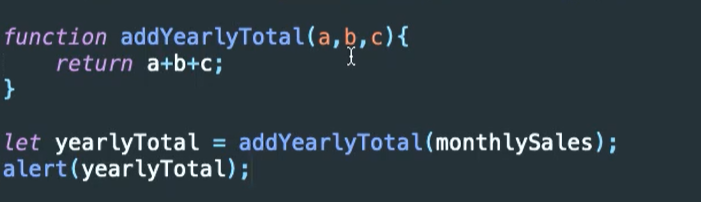


First one array.Of create a 3 spots for three numbers.

Second one Array(5) create an array with three places but each of them are empty.



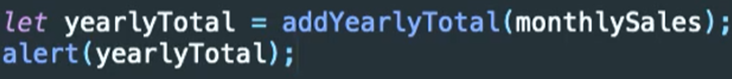
This feature works in the same way with string type.



Nothing special is a method which return a sum of 3 nums.



Our numbers looks like this above.



When dev calls this method in this way he get a



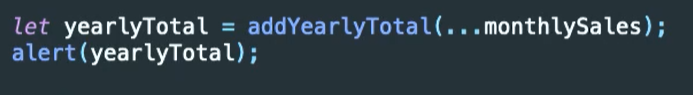
Because first arg is array second and third one are undefined.

Of course there is a option to make something like this

addYearlyTotal(Arr[0], Arr[1], Arr[2]), but we need a function which works in flexible way.

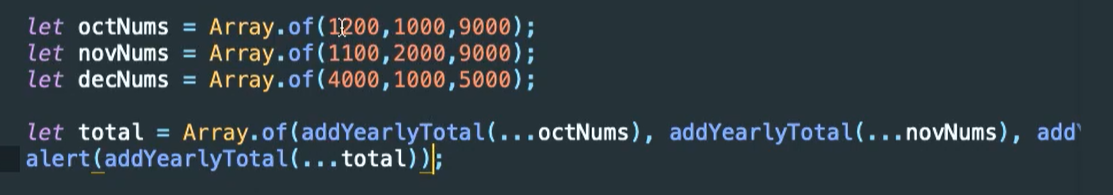
So the best practice is Linq SUM ☺. Nope it is JS

## Spread operator



It Is need a spread operator.

 now it works in proper way.



Another example there are a three examples of Array. From each of them are calculate a total for three months. In alert function whole arrays of sums is calculate for one big sum for all months.

## Find and FindIndex



Find method look for a element which is greater than 1000. When method found first element which was match to this statement, it is returned.

works in appropriate way.



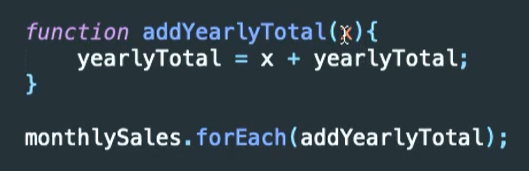
Another example works the same but, this method return an index of value which is adequate to statement. Obviously it is 9000 which is locate on first position Arr[1] = 9000.

## Fill



This method set all values in an array to params. In this example it is 0.

## Iteration

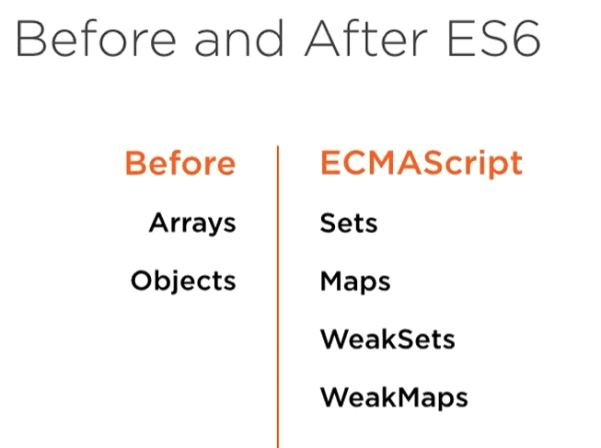


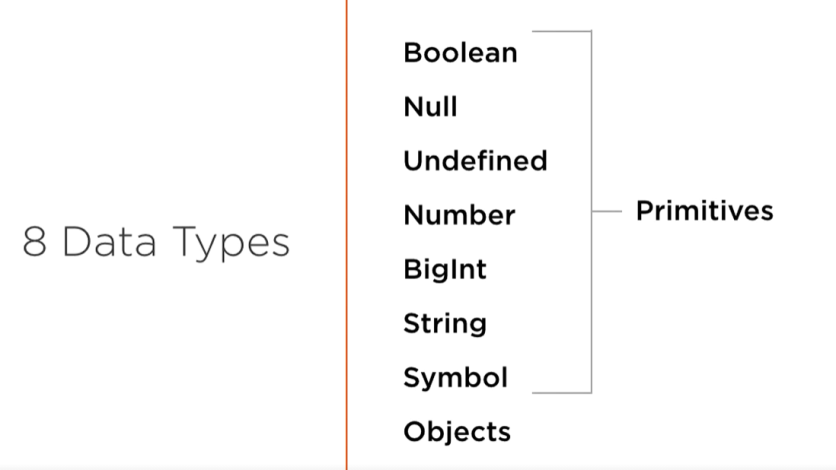
Yep it is much better than before.

# Data Collection

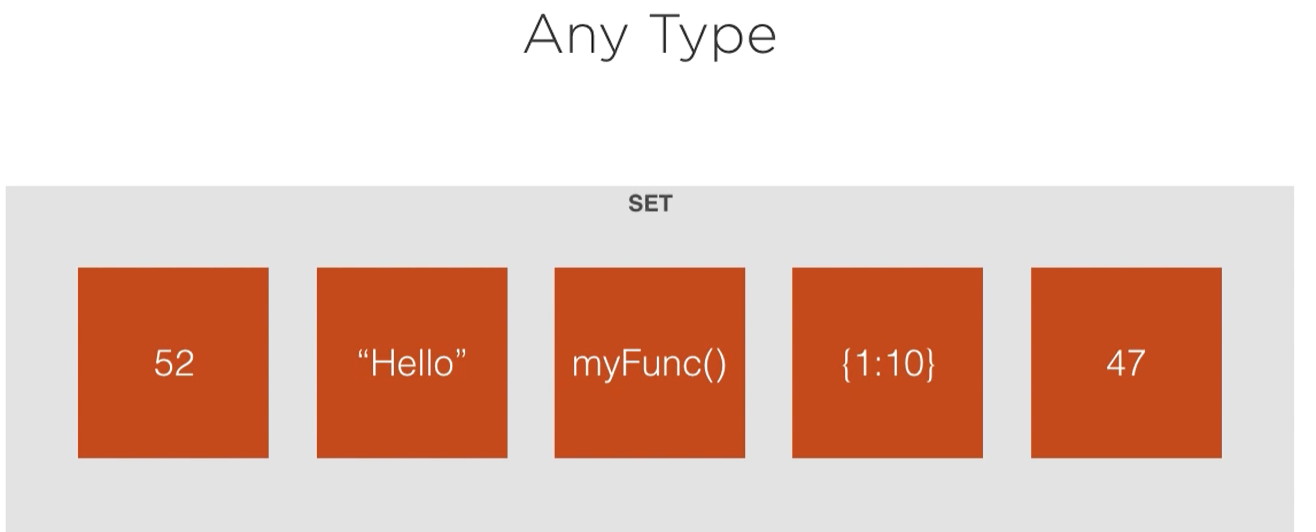
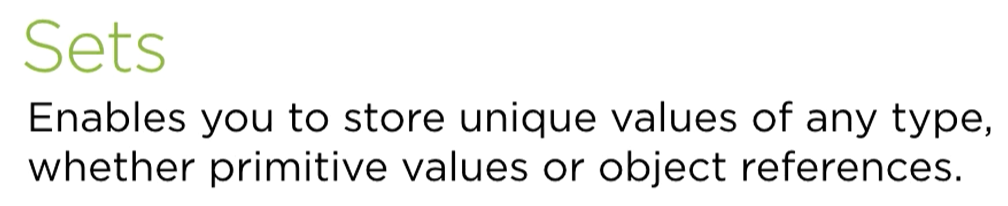
## Intro



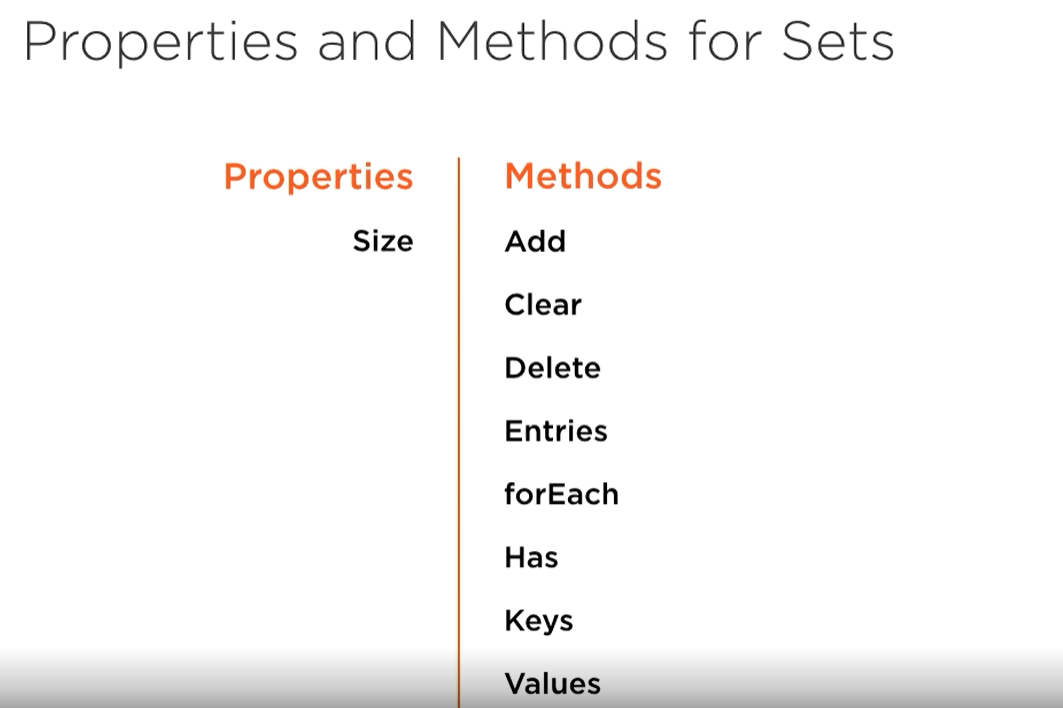




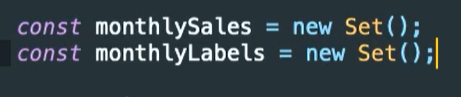
# Set



Yes this is a JS.



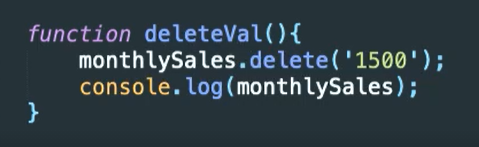
## Manipulate data in sets



To create set use new keyword.



To add a value to set use method add and pass a value.

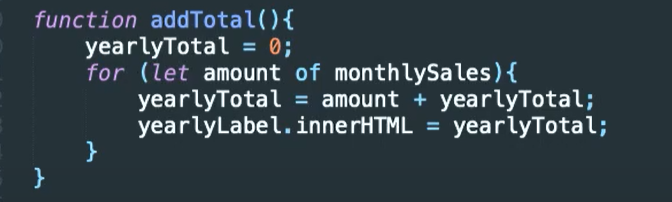




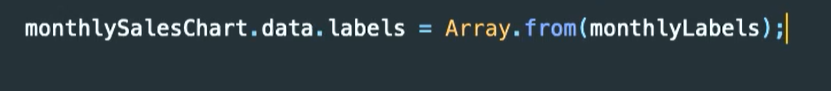
To set value can be add in this way.



Or this

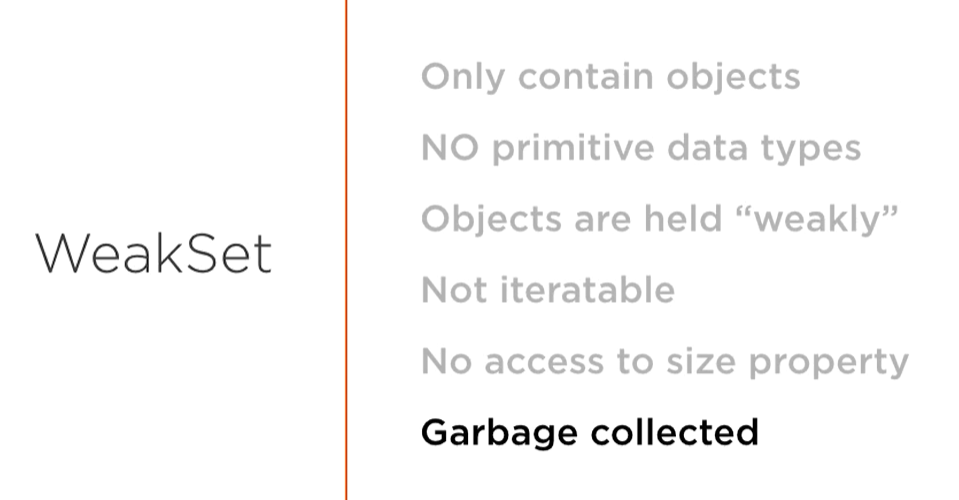


Sets are iterable

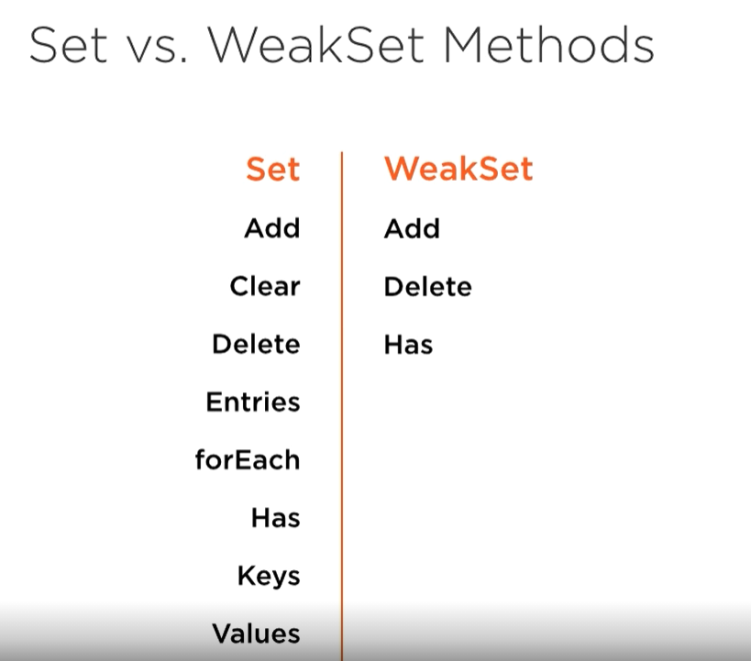


To make set as Array required is method Array.from(set) to do this operation.

## Weak Set

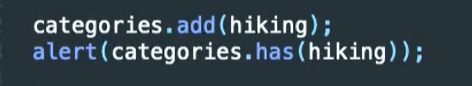






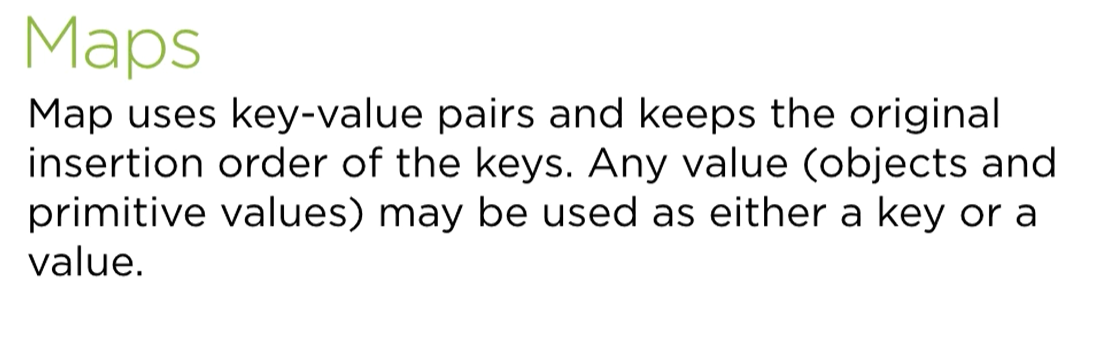


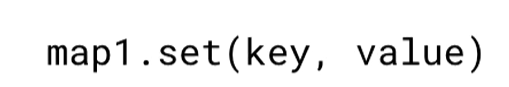
Creation



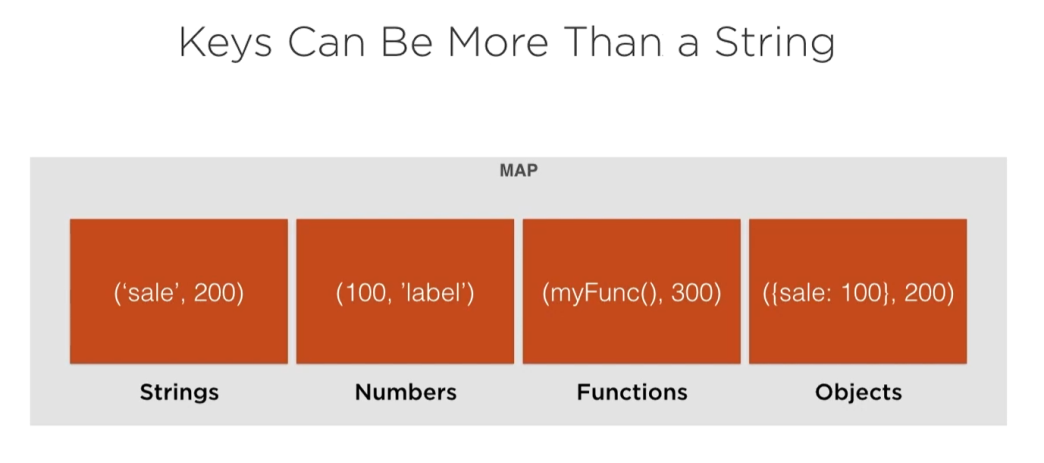
Has return boolean. Add naturally put object to weakset.

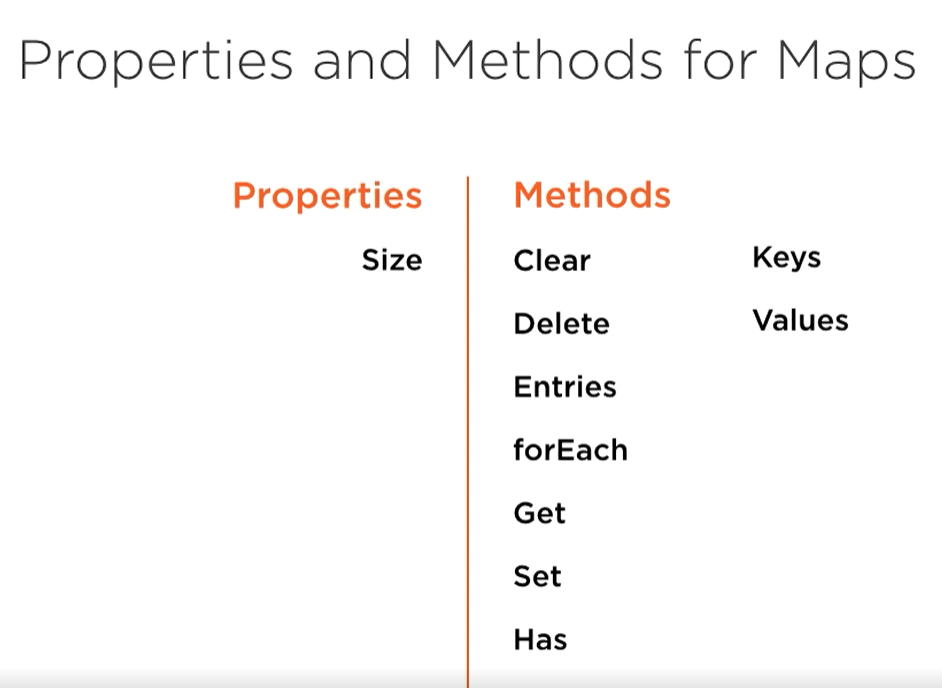
# Map





To create pair use map.ser(k,v);







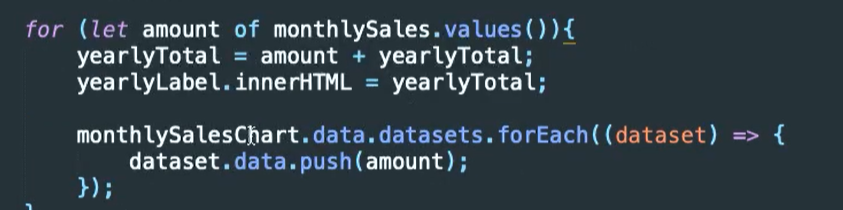
Create map and add new k, v pair.



Get value, required is a key.



To delete value need is a key.



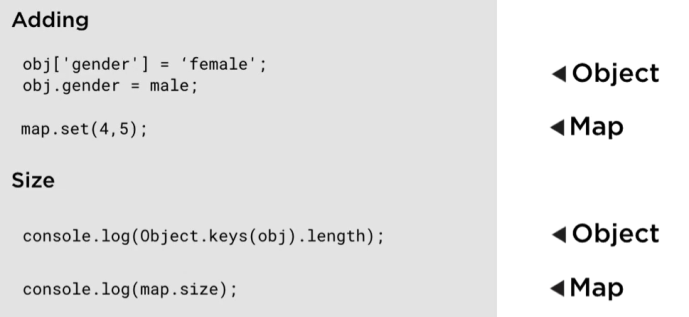
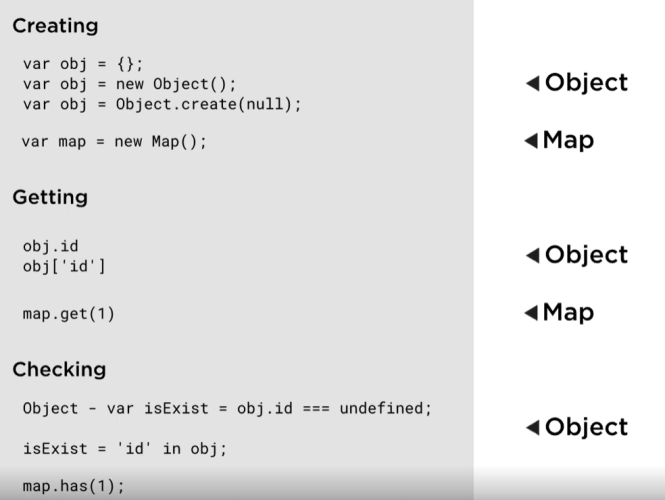
Iterate through map.

## Difference between Map and Object

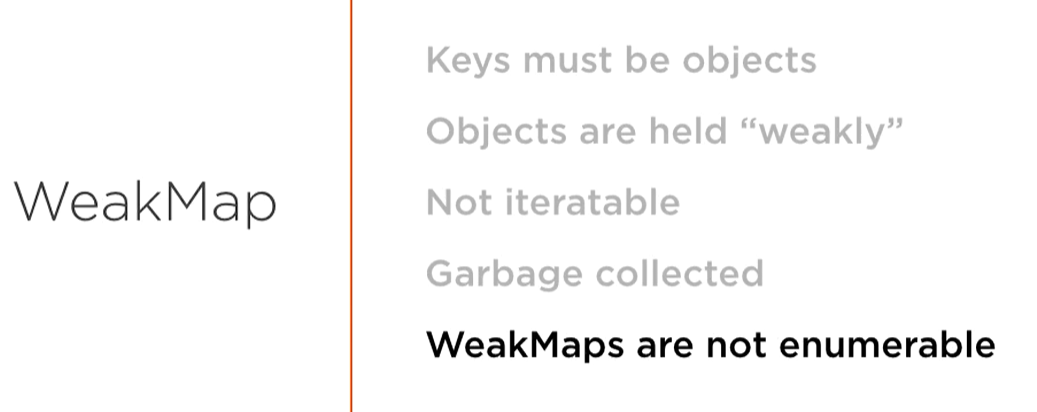
Faster iterate through map

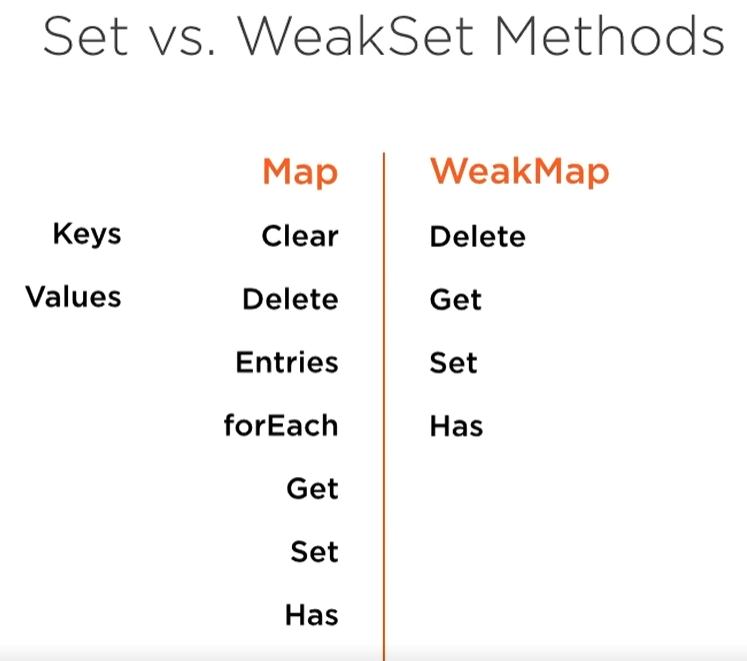
Element order in map

Map inherit after objects

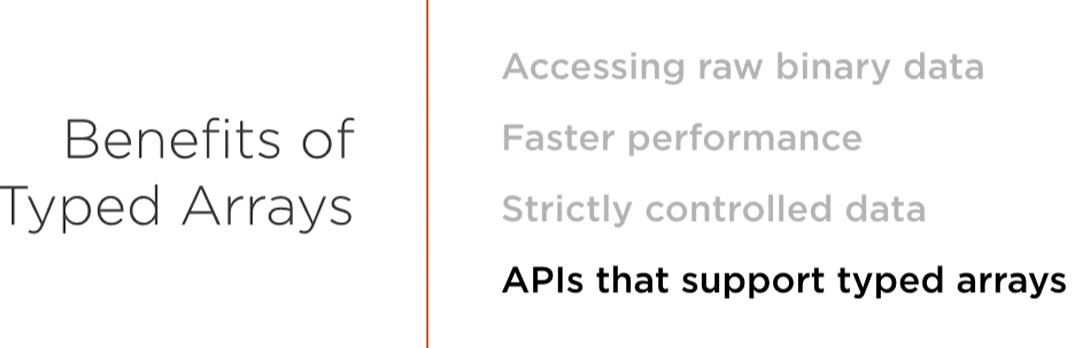


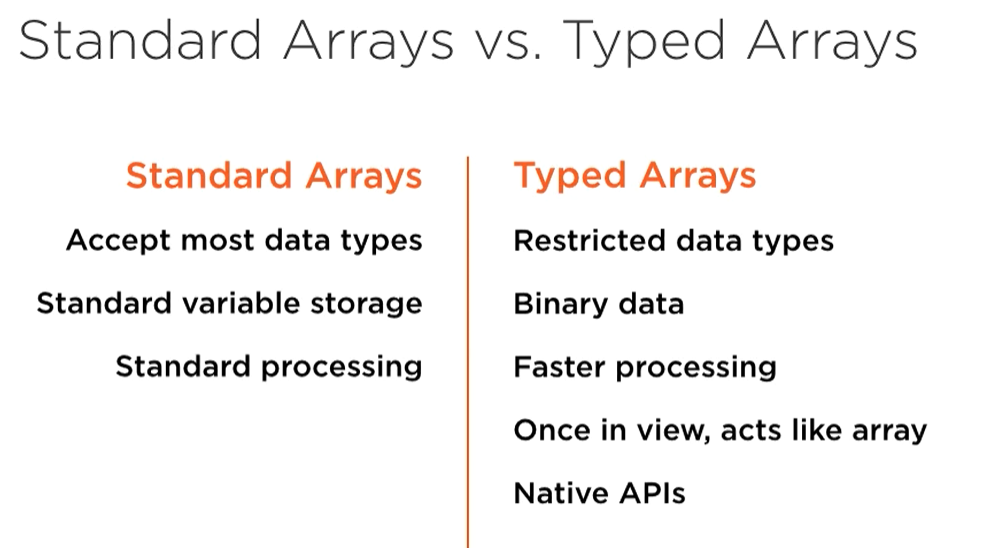
## Weak Map





## Typed Array





## Create Buffer



To create use ArrayBuffer class with byte length .



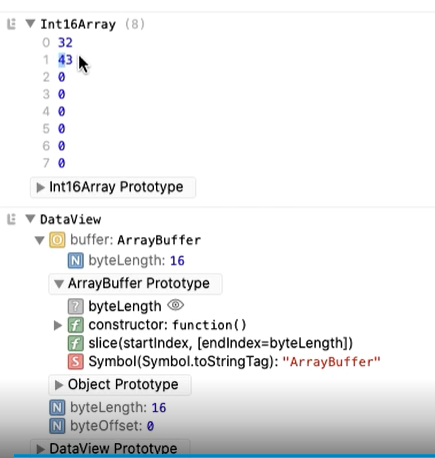
To create specific type array use TypeByteLength and add array buffer as argument.

 if value is unacceptable then in array is put the lowest num





Data view show information about particular data.





Set is use to add value on specific position

Get retrieve data from this position