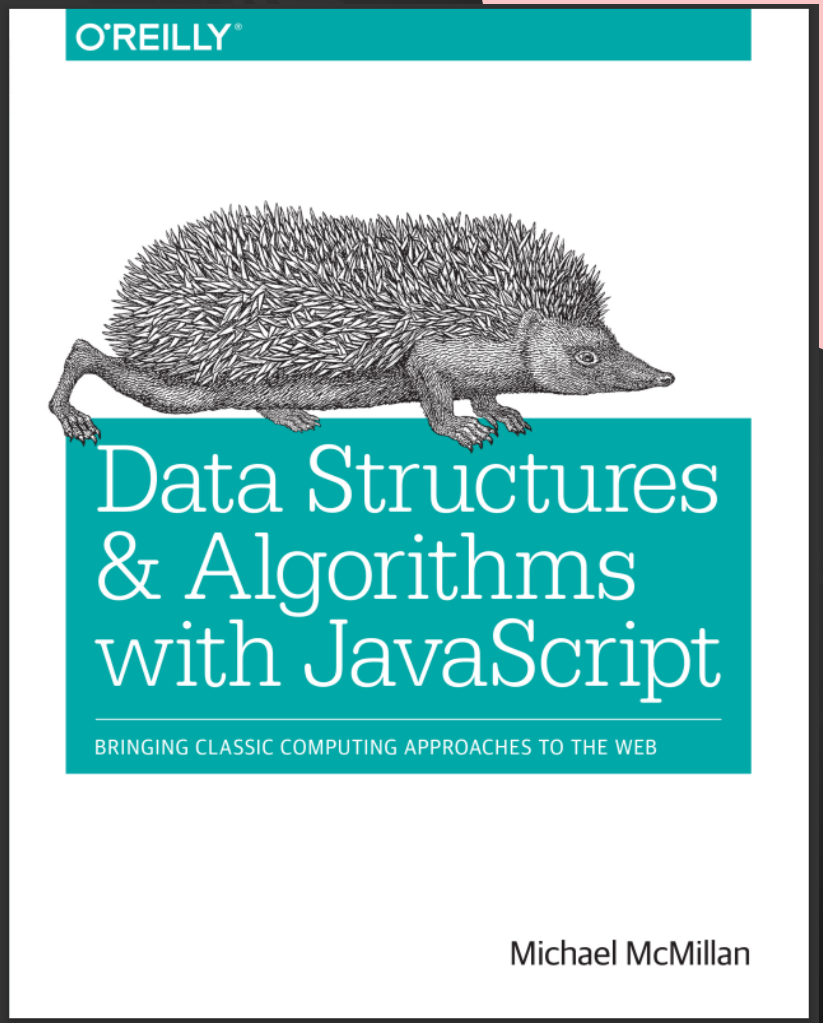
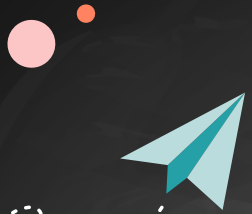
Chapter 2 (Arrays-Part 1)
Data Structures & Algorithms With javascript



Arrays Of Contents :-




Javascript Array Defined :-

- Linear collection of elements where the elements can be accessed via indices .

Creating Arrays :-

01

By declaring an array variable using the [ ] .

```
Var numbers = [];
```

02

Creating array by declare array variable with a set of element .

```
Var numbers = [1,2,3,4];
```

03

Creating an array by calling the array constructor.

```
Var numbers = new Array();
```

04

Creating an array by calling the array constructor with a set of element as arguments.

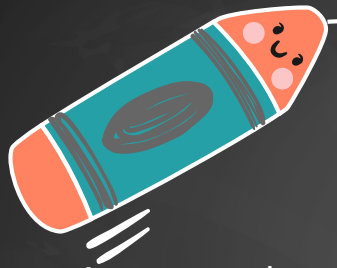
```
Var numbers = new Array(1,2,3,4);
```

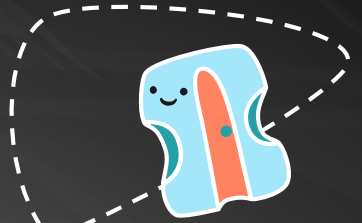
Notice :-

Javascript array elements do not all have to be of the same type .

Var objects = [1, “joe”, true, null]

Creating Arrays from String :-





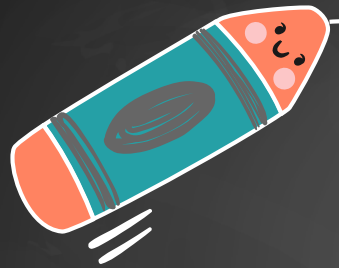
Arrays can be created as the result of calling the split() function on a string. This function breaks up a string at a common delimiter, such as a space for each word, and creates an array consisting of the individual parts of the string.

```
1  var sentence = "the quick brown fox jumped over";
2  var words = sentence.split(" ");
3  for (var i = 0; i < words.length; ++i) {
4   console.log("word " + i + ": " + words[i]);
5  }
6  console.log(words)
7  console.log(typeof(words))
```

Console ×

```
word 0: the
word 1: quick
word 2: brown
word 3: fox
word 4: jumped
word 5: over
▼ (6) ["the", "quick", "brown", "fox", "ju...]
   0: "the"
   1: "quick"
   2: "brown"
   3: "fox"
   4: "jumped"
   5: "over"
object
```

Aggregate Array Operations :-



Assign one array to another array but when you assign one array to another array , you are assigning a reference to the assigned that lead to make a change to the array original array , the change is reflected in the other array .

```
1   var nums = [];
2   for (var i = 0 ; i < 100 ; ++i)
3   { nums[i] = i+1 ; }
4   var samenums = nums ;
5   nums[0] = 400 ;
6   console.log(samenums[0]) ;
7
8
```

Console ×

```
400
```

Accessor Functions :-

JavaScript provides a set of functions you can use to access the elements of an array

01

Searching for a Value

One of the most commonly used accessor functions is indexOf() . the function returns the index position of the argument. If the argument is not found in the array, the function returns -1

02

String Representations of Arrays

There are two functions that return string representations of an array: join() and toString()

03

Creating New Arrays from Existing Arrays

There are two accessor functions that allow you create new arrays from existing arrays: concat() and splice().

- The concat() function allows you to put together two or more arrays to create a new array .
- the splice() function allows you to create a new array from a subset of an existing array .

shallow copy

shallow copy

is new array simply points to the original array's elements. A better alternative is to make a deep copy, so that each of the original array's elements is actually copied to the new array's elements. An effective way to do this is to create a function to perform the task :-

script.js

```
var nums = [ ] ;
for ( var i = 0; i < 100; ++i ) {
nums[i] = i+1;
}
var samenums = [ ] ;
//Copy ( nums , samenums) ;
samenums = Array.from(nums);
nums [0] = 400 ;
console.log(samenums[0]) ; // displays 1
```

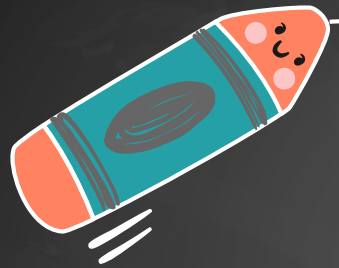
Console

```
1
```

Mutator Functions :-

- JavaScript has a set of mutator functions that allow you to modify the contents of an array without referencing the individual elements. These functions often make hard techniques easy .
- EX :- **Push() , Pop() , Shift() , Unshift() , ...**

Iterator Functions :-



Non–Array-Generating Iterator Functions

Not generate a new array .
they either perform an operation on each element of an array or generate a single value from an array .
(forEach(), every(), some(), reduce(), reduceRight())

Iterator Functions That Return a New Array

Iterator functions that return new arrays .
(map(), filter())