Array Object

An array can also be created using the Array object. The Array constructor can be passed.

- A numeric value that represents the size of the array or
- A list of comma separated values.

The following example shows how to create an array using this method.

Example

```
var arr_names:number[] = new Array(4)


for(var i = 0;i<arr_names.length;i++;) {
   arr_names[i] = i * 2
   console.log(arr_names[i])
}
```

On compiling, it will generate following JavaScript code.

```
//Generated by typescript 1.8.10
var arr_names = new Array(4);


for (var i = 0; i < arr_names.length; i++) {
   arr_names[i] = i * 2;
   console.log(arr_names[i]);
}
```

Its output is as follows –

```
0
2
4
6
```

Example: Array Constructor accepts comma separated values

```
var names:string[] = new Array("Mary","Tom","Jack","Jill")
```

```
for(var i = 0;i<names.length;i++) {

   console.log(names[i])

}
```

On compiling, it will generate following JavaScript code –

```
//Generated by typescript 1.8.10

var names = new Array("Mary", "Tom", "Jack", "Jill");

for (var i = 0; i < names.length; i++) {

   console.log(names[i]);

}
```

Its output is as follows –

```
Mary
Tom
Jack
Jill
```

Array Methods

A list of the methods of the Array object along with their description is given below.

S.No.	Method & Description
1.	**concat()** Returns a new array comprised of this array joined with other array(s) and/or value(s).
2.	**every()** Returns true if every element in this array satisfies the provided testing function.
3.	**filter()** Creates a new array with all of the elements of this array for which the provided filtering function returns true.

4.	**forEach()** Calls a function for each element in the array.
5.	**indexOf()** Returns the first (least) index of an element within the array equal to the specified value, or -1 if none is found.
6.	**join()** Joins all elements of an array into a string.
7.	**lastIndexOf()** Returns the last (greatest) index of an element within the array equal to the specified value, or -1 if none is found.
8.	**map()** Creates a new array with the results of calling a provided function on every element in this array.
9.	**pop()** Removes the last element from an array and returns that element.
10.	**push()** Adds one or more elements to the end of an array and returns the new length of the array.
11.	**reduce()** Apply a function simultaneously against two values of the array (from left-to-right) as to reduce it to a single value.
12.	**reduceRight()** Apply a function simultaneously against two values of the array (from

	right-to-left) as to reduce it to a single value.
13.	**reverse()** Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.
14.	**shift()** Removes the first element from an array and returns that element.
15.	**slice()** Extracts a section of an array and returns a new array.
16.	**some()** Returns true if at least one element in this array satisfies the provided testing function.
17.	**sort()** Sorts the elements of an array.
18.	**splice()** Adds and/or removes elements from an array.
19.	**toString()** Returns a string representing the array and its elements.
20.	**unshift()** Adds one or more elements to the front of an array and returns the new length of the array.

Array Destructuring

Refers to breaking up the structure of an entity. TypeScript supports destructuring when used in the context of an array.

Example

```
var arr:number[] = [12,13]

var[x,y] = arr

console.log(x)
```

```
console.log(y)
```

On compiling, it will generate following JavaScript code.

```
//Generated by typescript 1.8.10

var arr = [12, 13];

var x = arr[0], y = arr[1];

console.log(x);

console.log(y);
```

Its output is as follows –

```
12
13
```

Array Traversal using for...in loop

One can use the **for...in** loop to traverse through an array.

```
var j:any;

var nums:number[] = [1001,1002,1003,1004]


for(j in nums) {

   console.log(nums[j])

}
```

The loop performs an index based array traversal.

On compiling, it will generate following JavaScript code.

```
//Generated by typescript 1.8.10

var j;

var nums = [1001, 1002, 1003, 1004];


for (j in nums) {

   console.log(nums[j]);

}
```

The output of the above code is given below –

```
1001
1002
1003
1004
```

Arrays in TypeScript

TypeScript supports the following concepts in arrays –

S.No.	Concept & Description
1.	**Multi-dimensional arrays** TypeScript supports multidimensional arrays. The simplest form of the multidimensional array is the twodimensional array.
2.	**Passing arrays to functions** You can pass to the function a pointer to an array by specifying the array's name without an index.
3.	**Return array from functions** Allows a function to return an array