Arrays

Arrays are list-like objects whose prototype has methods to perform traversal and mutation operations.

Mutation means modifying the existing array where as Non-mutation means producing new array as the result of an operation.

The length of the array and types of its elements is not fixed. The length of array can be changed at any time and data can be stored at non-contiguous locations also in the array.

It gives an ordered collection.

Creating an array

An array can be created in two ways. One is by calling constructor of Array class.

Syntax:

let arr =new Array(element-1 , element-2 ,....., element-n);

Another by using literal like below

Syntax:

let arr=[element-1,element-2,…………,element-n];

The second one is more convenient way to create an array.

An array can store element of any type.

Ex: let arr=[10,"abc",{a:10,b:20}];

Array elements are numbered, starting with zero which is called index. Index of an array always starts with 0.

Ex:

```
let  cities=["Hyderabad","Chennai","bangolore"];

console.log(typeof cities); //prints object

console.log(cities[0]);//prints Hyderabad

console.log(cities[1]);//prints Chennai
```

Length of array

length property returns the number of elements of array.

Ex: console.log(cities.length);//prints 3

Methods of Array

The prototype of an array has the following useful methods.

➔ **push(element/elements)**
This method adds elements to end of array.
Ex:

```
let a=[10,20];

a.push(200,300);
console.log(a);//prints [10,20,200,300]
```

push() mutates array. That means it modifies original array.

➔ **pop()**
It removes element from end of array and returns it.
Ex:

```
let a=[10,20];
let removedData=a.pop();
console.log("array is ",a,"and removed element is ",removedData);
```

pop() mutates array.

➔ **unshift(elements)**
This method adds elements to beginning of array.
Ex:

```
let arr=[10,20];
arr.unshift(30,40,50)
console.log("array is ",arr);
```

unshift(elements) mutates array

➔ **shift()**

It removes element from beginning of array.
Ex:

```
let arr=[10,20];
let removedElement=arr.shift()
console.log("array is ",arr," and removed element is ",removedElement);
```

shift() mutates array

➔ **splice(start,deleteCount)**
This method removes and return array of elements from "start" index upto the value of "deleteCount"
Ex:

```
let arr=[10,20,30,40,50,60];
let removedElements=arr.splice(1,1);//removes one value from index 1
console.log("array is ",arr," and array of removed elements is ",removedElements);
```

➔ **splice(start , deleteCount , newElements)**
The other version of splice method can add new elements into the positions of removed elements.
Ex:

```
let arr=[10,20,30,40,50,60];
let removedElements=arr.splice(1,1,2000);//removes one value from index 1 and
                                //inserts 200 in that palce
console.log("array is ",arr," and array of removed elements is ",removedElements);
```

Control flow statements

These statements are used to control the flow of execution of application based on the situation.

The following are very useful control flow statements.

- Decision making statements
- Loop statements(Iteration statements)
- Switch statement

Decision making statements

These statements takes condition and based on outcome of that condition, the particular block of code can be executed and other can be skipped.

- If statement
- If-else statement
- If- else if statement

If statements:

Syntax:

```
If(condition)
{
        If-block
}
```

If the the given condition is true, then if-block will be executed, otherwise,if block execution will be skipped.

If-else statement:

Syntax:

```
If(condition)
{
If-block
}
else
{
Else block
}
```

If the condition is true , then if block can be executed and else block will be skipped.Otherwise, if block can be skipped and else block can be executed.

If-else-if statement:

This statement can be used to check multiple conditions.

Syntax:

```
If(condition-1)
{
If-block
}
Else if(condition-2)
{
}
Else if(condition-3)
{
}
……………
```

Loop statements

To execute,a instruction or set of instructions repeatedly for required number of times,loop statements can be used.

They are

- For loop
- While loop
- Do-while loop

For loop:

Syntax:

```
for(initialization; condition check;increment/decrement)
{
        Definition of for loop
}
```

While loop:

Syntax:

```
Initialization;
while(condition check)
{
Inc/dec
}
```

Do-while loop:

Syntax:

```
Initialization;

do{

Inc/dec

}while(condition);
```

Note: for and while wont execute even once,if the initial condition is wrong. But do-while loop can execute at least once even the condition is wrong.

Switch statement

This statement is used for direct selection.It receives choice as value and a specific case will be executed based on that choice.

Syntax:

```
switch(choice)
{
case 1:xxxxxxxxxxx
        Break;
case 2:xxxxxxxxxxx
        Break;
……………………..
………………………
default: xxxxxxxxxxxxxx
}
```

Default parameters

We know that, the default value of parameters of a method is "undefined". It may generate "NaN", if the method performs any mathematical operation for example. To avoid such kind of issues, it is possible to assign some other default values to parameters of a function.

Syntax:

```
function   function-name (variable = value)
        {
        }
```

Ex:

```
function   test (a=10)
{
        console.log(a);
}
```

When we call the above function without any argument, it will use default value of "a" as 10.

test(); //prints 10

If we call it by passing argument, the the default value replaced by argument.

test(100); prints 100

Rest parameter

This type of parameter can receive 0 to n of arguments and place them into an array.The parameter which is preceded with "…" operator is called rest parameter.

Ex:

```
function test(...a)
{
        Console.log(a);
}
```

test(); //prints []

test(10,20); //prints [10,20]