# JavaScript –ES6

Day 4

# Agenda

- New ES6 syntax
- Arrow Functions
- Destructuring
- ES6 Modules
- ES6 Classes
- Promises
- ES6 collections
- Array extensions
- Object extensions
- String extensions

# Day 3 -Recap

- ES6 Modules
- ES6 Classes
- Promises

# Day 4

- ES6 collections
- Array extensions
- Object extensions
- String extensions

# ES6 collections

- ES6 introduces two new data structures: Maps and Sets.

Maps − This data structure enables mapping a key to a value.

Sets − Sets are similar to arrays. However, sets do not encourage duplicates.

# Maps

The Map object is a simple key/value pair. Keys and values in a map may be primitive or objects.

- Following is the syntax for the same.

new Map([iterable])

The parameter iterable represents any iterable object whose elements comprise of a key/value pair. Maps are ordered, i.e. they traverse the elements in the order of their insertion.

# example

- var roles = new Map([
  ['r1', 'User'],
   ['r2', 'Guest'],
   ['r3', 'Admin'],
]);
console.log(roles.get('r2'))
2) var map = new Map();
map.set('name','Tekisky');
map.get('name');

```
let map1=new Map([['n0','sam'],['n1','abdullah']]);
    console.log(map1);
    console.log(map1.get('name'));
    map1.set('n2','sohail');
    console.log(map1.get('n2'));
     map1.clear();
     console.log(map1);
   map1.delete('name');
   console.log(map1);
   console.log(map1.size);
   console.log(map1.has('n12'));
  map1.forEach((value,key)=>{
    console.log(value,key);
  })
```

# Sets

- A set is an ES6 data structure. It is similar to an array with an exception that it cannot contain duplicates. In other words, it lets you store unique values. Sets support both primitive values and object references.
- Just like maps, sets are also ordered, i.e. elements are iterated in their insertion order. A set can be initialized using the following syntax.

```
var  set = new Set(['a','b','c','d','e']);

var iterator = set.entries();

console.log(iterator.next())
```

# Ternary Operator

- A ternary operator can be used to replace an if..else statement in certain situations. Before you learn about ternary operators, be sure to check the JavaScript if...else tutorial.
- condition ? expression1 : expression2
- The ternary operator evaluates the test condition.

- If the condition is true, expression1 is executed.
- If the condition is false, expression2 is executed.
- The ternary operator takes three operands, hence, the name ternary operator. It is also known as a conditional operator.

```
let x=31;
    //let result=(x%2==0)? 'even number' :'odd number'
    if(x%2==0){
        result='even number'
    }else{
        result='odd number'
    }
    console.log(result);
```

- Any Question

- Thank you