21/4/

# ECMA-6

It is a trademark scripting language specification standardized by ecma internation org.

* It was created to standardized java script so as to foster multiple implementations.

* ECMA 6 is also called as ECMA 2019 which is released after a long gap since 2009

* This release update add significant new syntax for writing complex applications including classes and modules etc.

* The other new features include iterators for of looping control structures python stype generators, arrow functions, Binary

data, type array, collections, promises etc

→ difference b/w let, const and var keywords.

Var Keyword: variable gets created using var keyword becomes ~~do~~ module scope

Eg: function display () {

```
    var i;
    if (true) {
    ...
    // i can be accessable.
    ...
    var b = 90;
    }
    ...
    // i and b can be accessable here.
}
// Outside of function i and b cannot be accessable.
```

let Keyword: using let keyword we could able to create block scoped variables.

The variables being created using let keyword can be accessable under a particular block cannot be accessable outside the block.

Eg: function display() {

```
    let i = 44;
    if (true) {
      ...
      // i can be accessable.
      ...
      let b = 90;
    }
    ...
    // only i can be accessable here, not b
}
// outside of function i and b cannot be accessable
```

Const Keyword: Variables been declared using const, once we initialize the variable we couldn't able to change the values in it (final variables)

For Each looping Control Structure

a pre-defined looping control structure we could able to iterate over an array which takes a call back method and throws the corresponding value and index value.

Syntax:

```
<array>.forEach(function(value, index){
  ...
```

});

Eg: var a = [67, 88, 9, 9];

a.forEach (function (value, index){

console.log(value);

console.log(index);

});

22/4/20

## Java Script Arrow functions

from ecma-6 to define a function within javascript we wont be using the function keyword but we can declare the function just by using arrow symbols("=>")

Syntax:

var <funname> = (<optional params>)
=> {

.....

// body of function...

}

Eg: var addValues = () => {

...

}

var mulValues = (a, b) => {

....

}

The main intention of using arrow functions is the arrow function holds current "this" operator even the particular function gets called after some time.

## Function parameters with default values

from ECMA-6 javascript supports a feature of having default values for formal parameters which will be considered automatically if there the corresponding values were not being passed for actual parameters.

Syntax:

```
function <funcname> (param1 = value1,
                     param2 = value2, ....) {

    ....
    ....

}
// calling function
funcname(value1); // value 2 is been taken
                     from default params.
```

Eg:

```
function addvalues (a, b = 89, c = 90) {
    ....
    ....
}
addvalues (23,45);
```

## Extended parameter Handling

"..." is a extended operator being supported in js using which all the parameters which are not been handled will be automatically catched in the form of a array.

Eg:
```
function addValues(a, b, ...c) {
    .....
}
addValues(23, 44, 666, 7, 28, 32);
```

Note:
Any object that is not holding any data is called a "Null"

## Array Assigned Values

If multiple values need to be assigned at a time we could able to declare the values under an array and assign array of values to it.

Eg: `[a, b, c] = [34, 56, 89];`

## Template literals or Template Strings

using template literals we could able to replace part of a string with given value.

Eg:
```
Var name = "Raj";
Var age = 60;
```

Var msg = `user name is ${name} and ur age is ${age}`;

23/4/20 Sets and ~~Arrays~~ Maps data structure

The two pre-defined data structures been supported from ecma-6.

Set data structure:

It is almost like array data structure used to hold group of relative item to identify the content with same variable name. the only difference b/w sets and arrays are ~~as~~ set accept only unique values, doesn't allow duplicate values

Syntax:

Initializing a set

Var s = new Set(); // creates a set data-structure

following are pre-defined methods can be applied on set data structure

- .add (`<value>`) // adds value to set
- set - returns total number of values in a set
- values() - returns values within set
- has ("value") - returns true/false if the provided value exist in set.
- clear() - Removes all values from set

## "for of" looping control structure

a pre-defined looping control structure through which we could able to iterate over a set of or map data structures

Syntax:

```
for (var temp of set.values()) {
    ....
    ....
}
```

- delete("value") – deletes corresponding value from set.

## Map data structure:

a pre-defined data structure almost like a json object used to hold the data in the form of key value pairs

Syntax:

```
var details = new Map();
```

following are predefined methods can be applied on map

- set("key", "value") - To add value along with corresponding key.
- get("key") - Returns corresponding value for a particular key
- entries() - return values within a map
- clear() - clears data from map
- delete("key") - deletes corresponding key and its value from map. et

Syntax to iterate over map DS

```
for (let [key, value] of map.entries()) {
    ....
    ....
}
```

Exponentiation operator:

"**" is a new exponentiation operator supported from ecma-6 through which we could able to find the power of numbers.

Objects rest (or) spread properties:

Same like extended parameter handling

Object Enchanced properties

Java Script classes

From ECMA-6 java script supports a keyword called class through which we could able to create classes in java script

25/4/20

Java Script classes (ECMA-6)

"class" is a pre-defined keyword through which we could able to create classes in java script.

* classes are used to create the basic structure of an object or skeleton of an object

(data members and

* for a single class we can create any no. of objects.
* "new" is an pre-defined keyword through which we could able to instantiate a class in javascript
* "constructor" is an pre-defined keyword through which we could able to add a constructor within a javascript class.
* constructor gets invoked automatically while we instantiate the particular class
* "super" is a pre-defined keyword through which we can invoke constructor of a parent class through child class.
* In javascript we cannot override constructor method

## Java Script promises

a new topic been supported from ECMA-6, promise is an object which represents the eventual successful compilation or failure of a asynchronous operation and its corresponding value.

* Every promise takes a resolver and rejection call back function which gets invoked based on success or error
* "promise" is a pre-defined class through which we can create any no. of promises

within a single application

Syntax:

```
Var promise = new Promise (function
                (successResolvercallback, rejectorcall
                                          back){
})
```

invoking promise:

```
method (<optional param>). then (successcallback).
                          catch (errorcallback);
```

New pre-defined string Methods supported in ECMA-6

* str.startswith ("msg", index) // returns true/ false if the provided msg is at given index position

* str.endswith ("msg", index) // return true/ false if the string ends with given msg at provided index

* str.includes ("hello"); // returns true if string is included.

* str.includes ("msg", 1); // returns if the msg is existed at given position.