1.Java script is Case sensitive

2.let introduced in EC5

3.const introduced in EC6

6.var was there before

Objects

Objects are variables too. But objects can contain many values. The values are written as **name:value** pairs (name and value separated by a colon).

const car = {type:"Fiat", model:"500", color:"white"};

or An object definition can span multiple lines:

const person = {  
  firstName: "John",  
  lastName: "Doe",  
  age: 50,  
  eyeColor: "blue"  
};

Objects can also have **methods**.

const person = {  
  firstName: "John",  
  lastName : "Doe",  
  id       : 5566,  
  fullName : function() {  
    return this.firstName + " " + this.lastName;  
  }  
};

**String**

let x = "John";  
let y = new String("John");

Do not create Strings objects.

The new keyword complicates the code and slows down execution speed.

String objects can produce unexpected results:

Note the difference between (x==y) and (x===y).

Extracting String Parts

There are 3 methods for extracting a part of a string:

* slice(*start*, *end*)
* substring(*start*, *end*)
* substr(*start*, *length*)

**Arrays**

|  |  |
| --- | --- |
| [parse()](https://www.w3schools.com/jsref/jsref_parse_json.asp) | Parses a JSON string and returns a JavaScript object |
| [stringify()](https://www.w3schools.com/jsref/jsref_stringify.asp) | Convert a JavaScript object to a JSON string |

const fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.sort();

console.log(fruits)

const fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.sort().reverse();

console.log(fruits)

By default, the sort() function sorts values as **strings**.

However, if numbers are sorted as strings, "25" is bigger than "100", because "2" is bigger than "1".

Because of this, the sort() method will produce incorrect result when sorting numbers.

You can fix this by providing a **compare function**:

Program-1

const points = [40, 100, 1, 5, 25, 10];

points.sort(function(a,b){

    return b-a

})

console.log(points[0])

program-2:

const cars = [

    {type:"Volvo", year:2016},

    {type:"Saab", year:2001},

    {type:"BMW", year:2010}

  ];

  cars.sort(function(a,b){

    return a.year - b.year

  })

  console.log(cars)

## The Array forEach() Method

program-2:

## print even nos

const numbers = [45, 4, 9, 16, 25];

var sum ="";

numbers.forEach(function(item){

    if(item%2===0)

    {

       // console.log("even no are "+item)

        sum=sum+" "+item

    }

})

console.log(sum)

const numbers = [45, 4, 9, 16, 25];

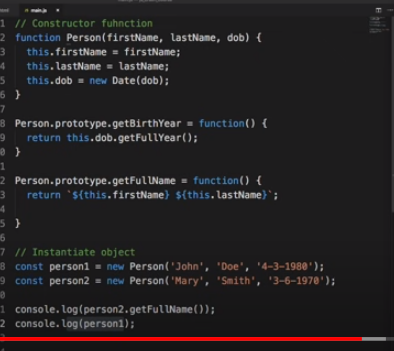
numbers.filter(function(num){

    return num%2==0

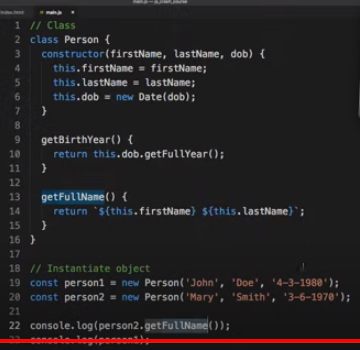
})

console.log(numbers)

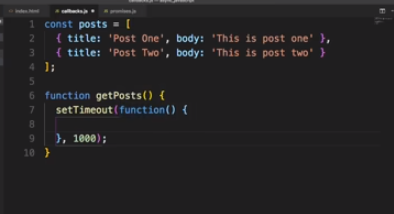
**EC5 -syntax**

****

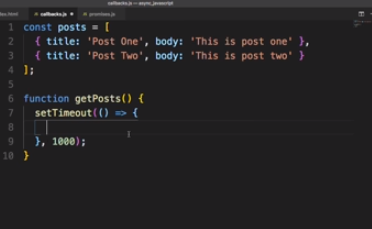
**ES6—syntax**

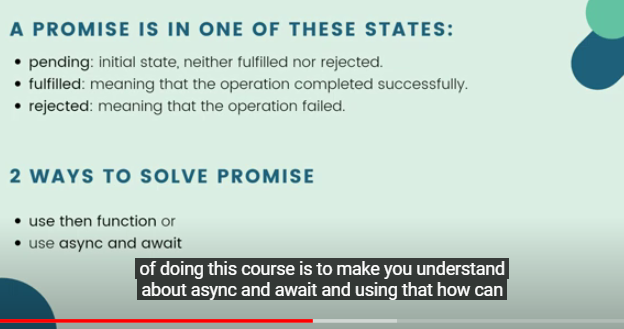
****

**EC-5**

****

**ES-6**

****

****

How to handle promise in ES-5 (with then function)

How to handle promise in ES-6(with async and await)

describe("Learn protractor",function(){

    it("Launch Browser",function()

    {

        browser.get("wwww.google.com")

        browser.getTitle().then(function(title){

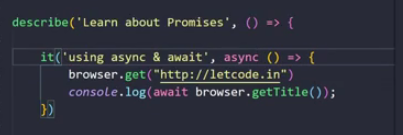
            console.log(`Title is {title}`)

        })

    });

});

ASync and await in ES6



const points = [40, 100, 1, 5, 25, 10];

let largest = points[0]

let i;

for(i=1;i<points.length;i++)

{

    if(points[i]>largest)

    {

        largest=points[i]

    }

}

console.log(largest)

const points = [40, 100, 1, 5, 25, 10];

points.sort(function(a,b){

    return b-a

})

console.log(points[0])

const cars = [

    {type:"Volvo", year:2016},

    {type:"Saab", year:2001},

    {type:"BMW", year:2010}

  ];

  cars.sort(function(a,b){

    return a.year - b.year

  })

  console.log(cars)

const numbers = [45, 4, 9, 16, 25];

var sum ="";

numbers.forEach(function(item){

    if(item%2===0)

    {

       // console.log("even no are "+item)

        sum=sum+" "+item

    }

})

console.log(sum)

const numbers = [45, 4, 9, 16, 25];

var evens = numbers.filter(function(num){

    return num%2===0

})

console.log(evens)

const numbers = [45, 4, 9, 16, 25];

var doubleEle = numbers.map(function(num){

    return num\*2

})

console.log(doubleEle)

const numbers = [45, 4, 9, 16, 25];

var greater=numbers.filter(function(num){

    return num>18

})

console.log(greater)

const numbers = [45, 4, 9, 16, 25];

let  total=1000

var sum=numbers.reduce(function(total,num){

    return total+num

})

console.log(sum)