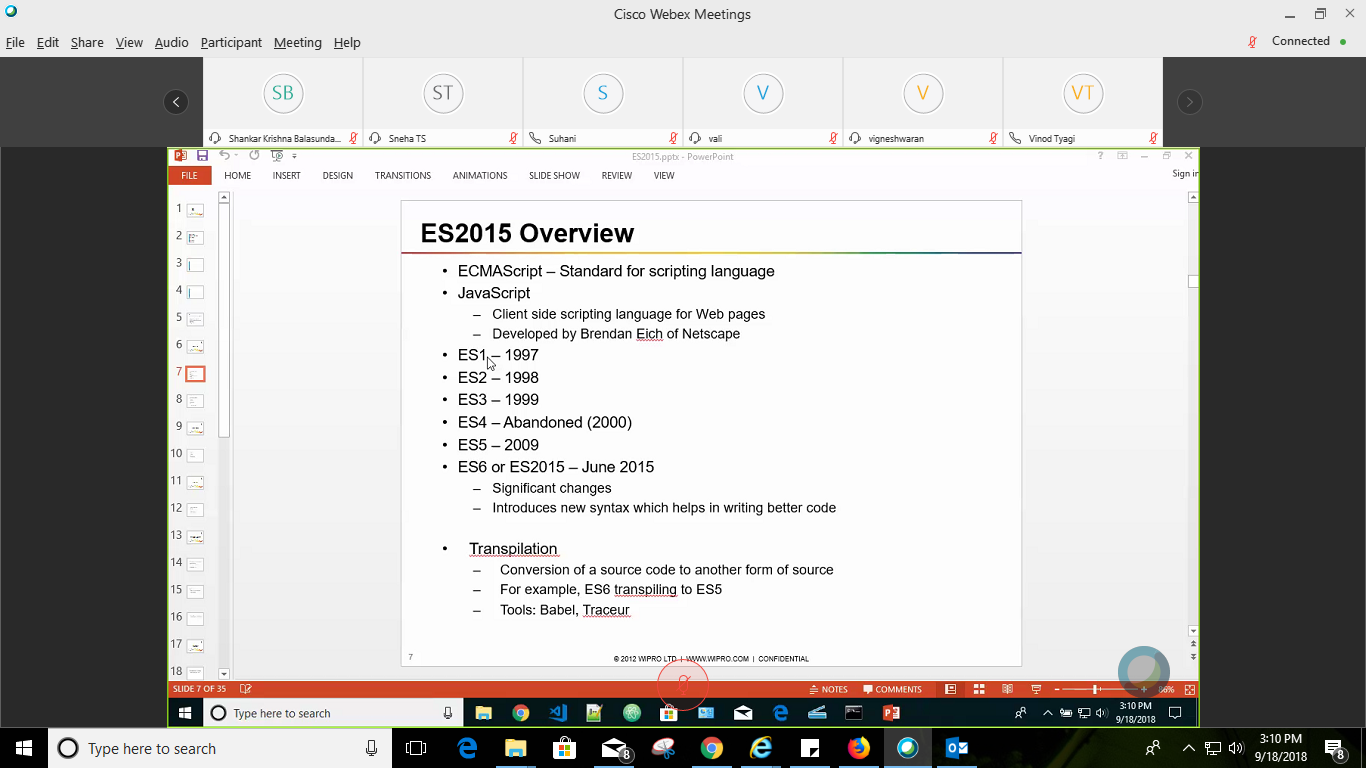
**Ecma Script TimeLine**

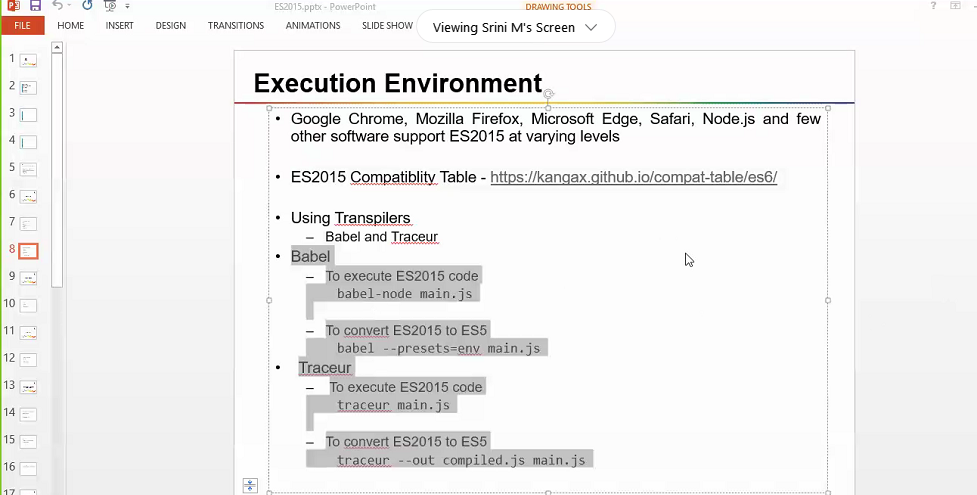


If ES4 was implemented we could have ES6 standards already implemented but companies didn’t agree for the same.

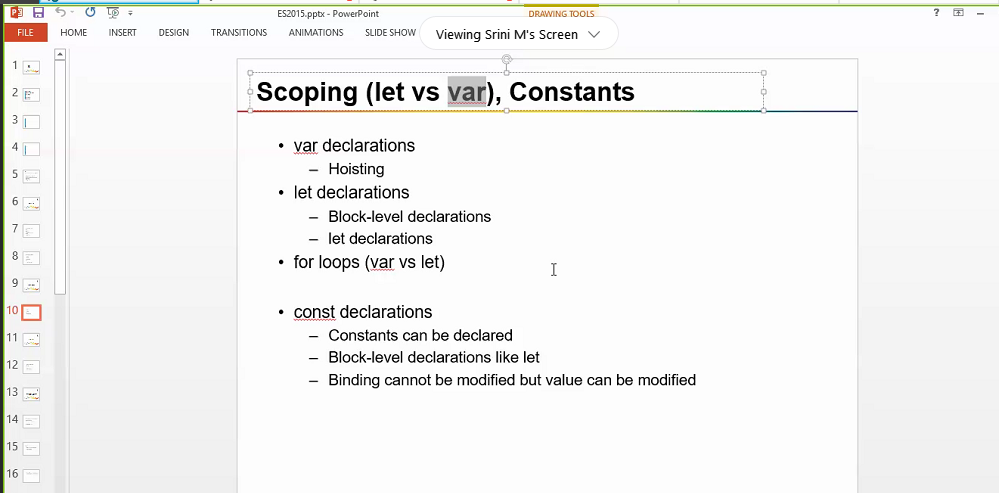
Syntactic sugar: New changes in syntax which help to write better code.

Some tools like Babel and Traceur you can convert es6 to ec5 code so that it can run on older browsers.

**Transpilation: to Convert one type of code to other.(standards)**

**How to run ES6 in old browsers**

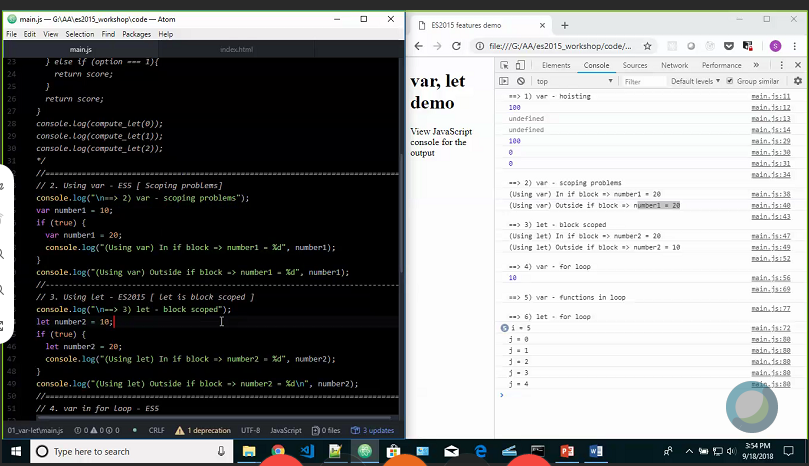
**The Difference between VAR AND LET**



**Hoisting Problem**

Where ever you declare a variable it gets moved to the top.

So issue like runtime errors happen variable not defined.



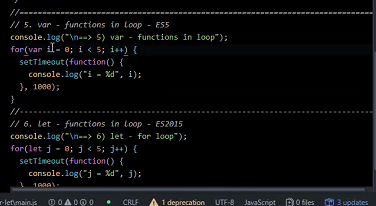
Block level scoping problem solved using **LE**

**For Loop problem**

Using Var we can access variable outside the loop.

But it doesn’t happen with **let.**

**Var-functions in loop**



Here we expect i=1,2,,3,4,5 but it comes as 5 each time because it behaves as hoisted variable.

So For async functions we use let

So j=1,2,3,4,5

**Converting es6 to es5 using babel**



Here we have to install babel dependencies too. And main.js is converted into es5-babel.js

During july 2015 we had to use these things

BABEL is more readable than traceur(its for experimental purpose)

**Const in ES6**

If we declare a variable as constant we cannot define any other value to it.

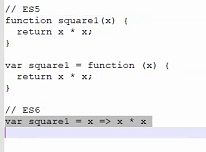
Const is a constant reference not a constant variable.

Eg const day={};//a constant object

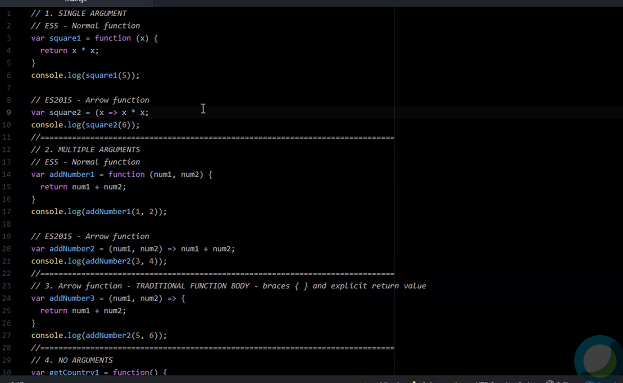
And we can modify day.first=”Monday”;

But we cannot do day=Monday;//this is an error i.e we cannot modify a reference

**Moreover same like let Const has a block scope.**



**Fat arrow function/Closures/Lambda Functions**



**No argument**

Var getCountry= ()=>”india”;

**No argument no body function**

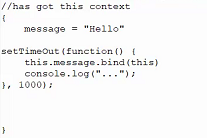
Var No=() => { };// generally useful for dummy test cases

**Returning an Object literal**

Var GetUserId= id =>({id:32, Name: “harsh”} ); //wrapping it in () paranthesis help us to know that it is an object not a function body.

Arrow function does not have the concept of inner function so,

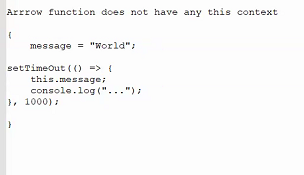
In traditional way inner function has **this** context.



Here we are binding outside message to inner message

Using this.message.bind(this)

For arrow function as we don’t have this context so we can use directly.



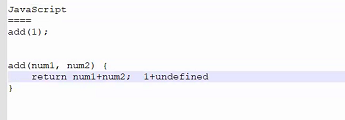


Here we have to store this wrt to outside display we need to store it in self var.

**ES6 way fat Arrow**

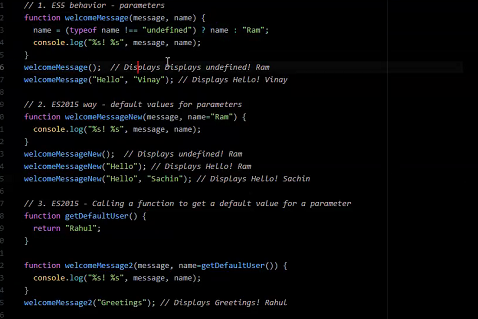


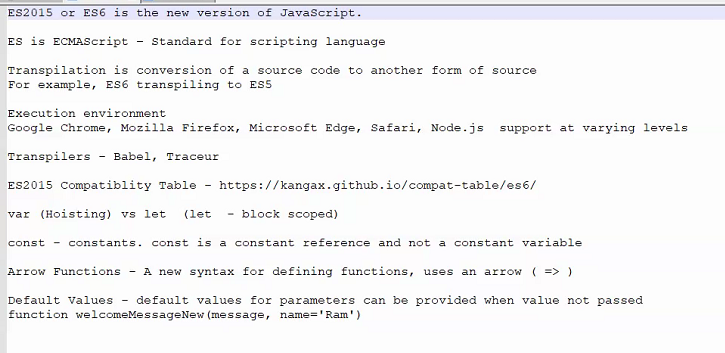
**Default Arguments Concept**



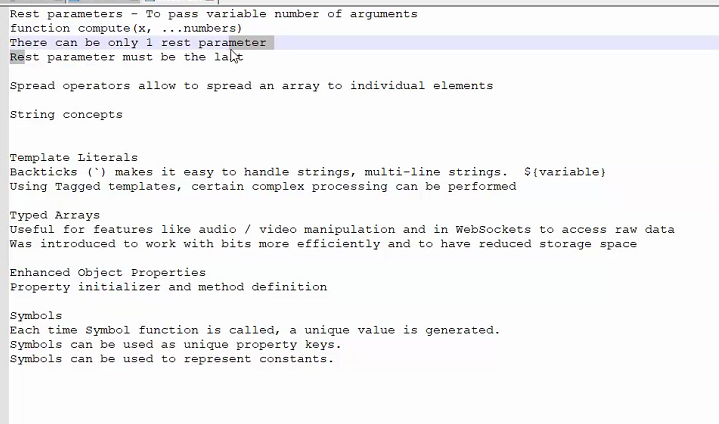
Here as 2nd argument is not passed so in runtime code will break.

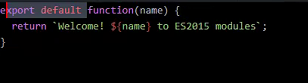
To solve this problem we have ES5 way vs Es6 way(just put it in argument itself or use a function to add value as in eg. 3):-





**Recap day 2**





directly add variable argument to use in ${ varname}

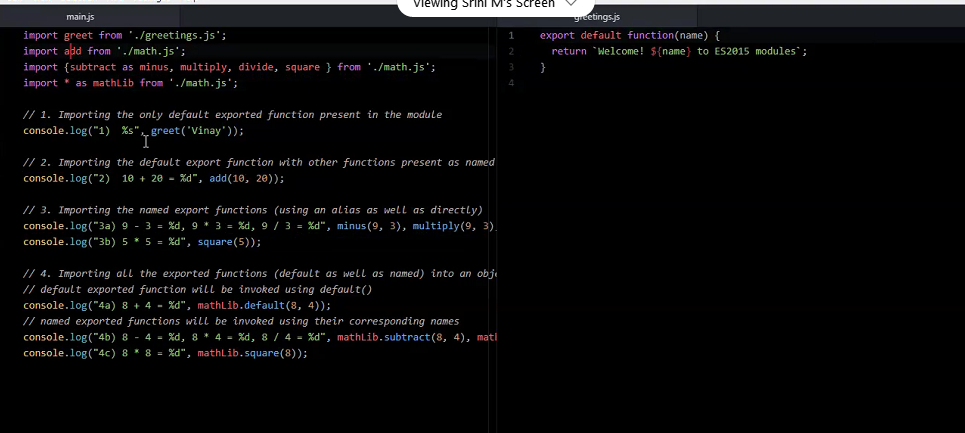
**Modules**

Webpacks :

We need to give a starting point to webpack and it will find all the relevant files and add them to a single file which can be directly added to index.js

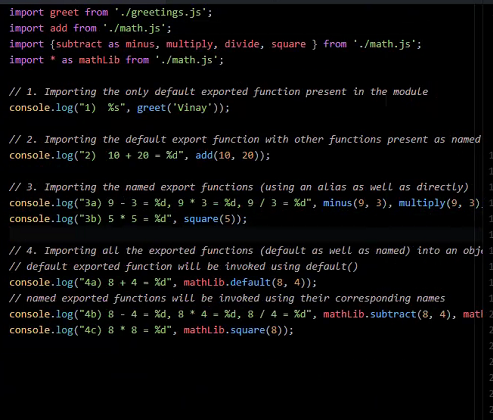
To generate single file do webpack –mode=development //if you don’t add dev it will automatically add it to production mode i.e the code will be more compressed here

Mutiple ways of export and import



Multiple ways of exporting:

1. One function can be default exported.
2. We can do export{function name}
3. While declaring function we can directly write export
4. Moreover we can use alias name in export like export(sq as square)

we 

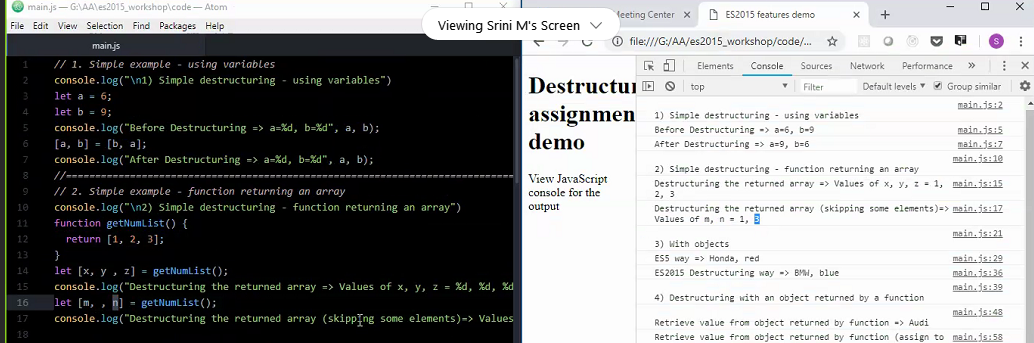
we can import either individually or maybe use **import \* to import all the functions.**

**Like here in example mathlib.default is used because we don’t have a name for default function**

**Destructuring**

This is way of destructuring breaking arrays objects and stuff.

A means to assign values from arrays and objects in variables.



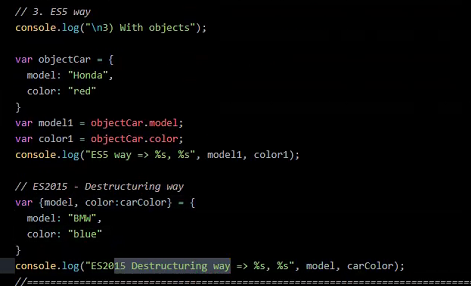
Here we swapped a,b in eg 1

And in eg 2 we get value from function directly in variables.

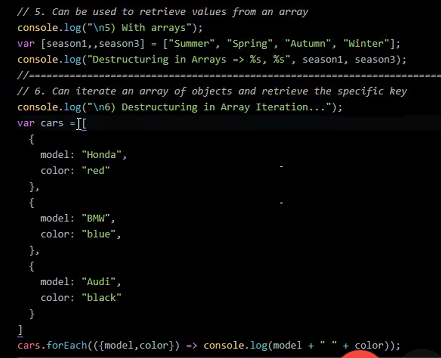
In this example we have used key:variable pairs like color:carcolor //key:var

And to copy we need to add value to key like color:”blue”

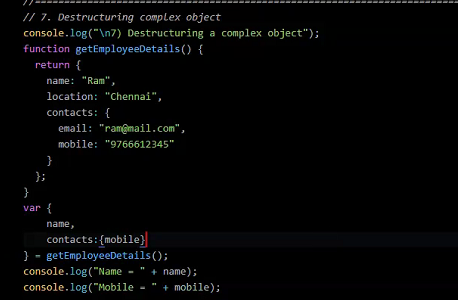
And to use it we need to use variable linked to the key



For arrays and objects eg here we used forEach to get all values of cars array.



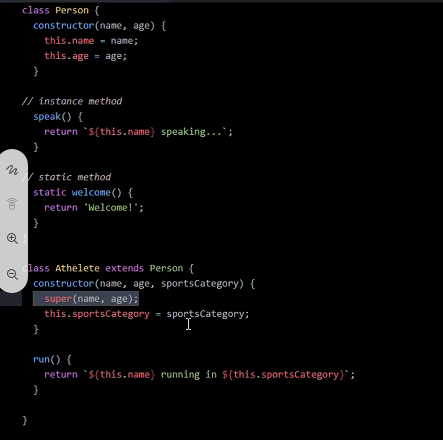
**Destructuring Complex data object**



Here var is created from copying from main function we just create same structure and remove values

If we need only email not mobile we don’t need to add that.

**Concept of CLASSES**



**Here we have same concepts like constructors instance methods functions inheritance etc.**

**Static methods can be called directly from class name**

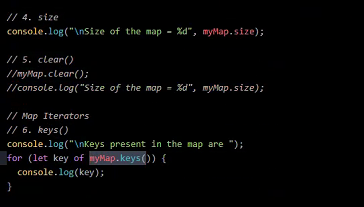
**ES5 way of implementing**

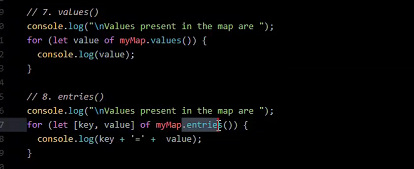
Prototyping: adding additional methods to class

Athlete.prototype.runner=function(){}

**MAPS AND SETS IN JS**



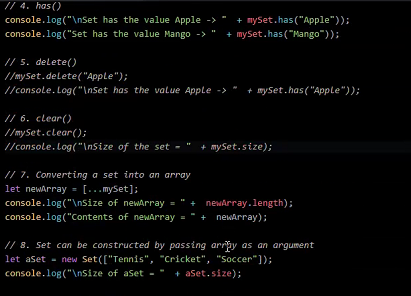


here we are using destructing to get key value pair to get values with keys

**SETS**

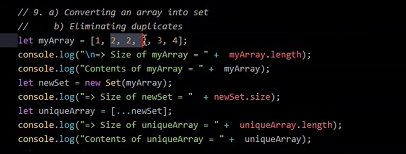


**Here important is in eg 3b we used entries here we don’t have keys but still we get key=value**



**Here we can set array as argument to create a new set.**

**Also we use spreading here to convert set to array […mySet] here … is spread operator**

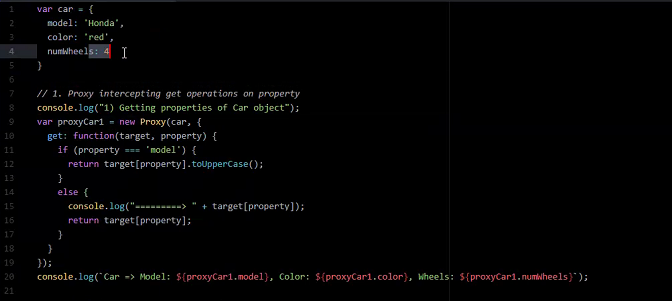


Here we convert duplicated array to a unique array

**PROXY(ONLY es6 concept)**

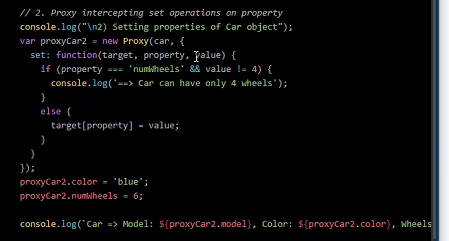
**It is a concept to filter out data. Or like real proxy we don’t expose our data directly it is a middleware thing which process in between**

**GET METHOD/ access value**



Here for the proxycar1 object new memory is created its just a wrapper for car object.

Here target is referring to object and property refer to car object property.



**SET method/ used for validation**

**Promises**

