ES6 features

1. Let/const
2. Objects
3. This
4. Arrow Funcs
5. Destructuring – map, filter and
6. Spread and rest
7. Classes
8. Modules
9. **Let ,Var and Cosnt:**

**Var** declared variable accessible upto function level

**Let** is accessible inside the block.

**Const** block scope access but reassign is not possible.

1. **Object**:

Const person = {name:’john’,walk: function() {}} in es6 Const person = {name:’’,walk() {}}

-collection of key value pair

-person.walk(); or assignment person[‘name’] = ‘john’;

Person.name=’change’;

1. **this**:

-always return reference of current object

Const person = {

Name:”M”,

Walk(){console.log(this);

}

};

person.walk();//object

Const walk = person.walk.bind(person);

walk();//undefined

-function are also objects in javascript.

1. **Arrow Function:**

Const square = function(number){return number \* number;}

Const square = number => { return number \* number;}

Const jobs = [

{id: 1, isActive: true},

{id: 2, isActive: true}

]

Const activejobs = job.filter(function(job){return job.isActive;});

Const activejobs = job.filter(job => job.isActive);

1. **Arrow Function and this :**

-arrow function don’t rebind this keyword

Const person = {

Talk(){

setTimeout(function(){console.log(“this”,this);

},1000);

}

};

Person.talk();//window object

setTimeout is standalone function as it not part of any object so this return window object

callback function overwrite the strict mode

Const person = {

Talk(){

Var self = this;

setTimeout(function(){console.log(“this”,self);

},1000);

}

};

Person.talk();//show person object

Const person = {

Talk(){

setTimeout(()=>{console.log(“this”,this);

},1000);

}

};

Person.talk();//show person object

1. Map:

Const color = [‘red’,’blue’,’green’];

Const items = Color.map(function(color){return ‘<li’+color+’</li>’;});

1. Object Destructuring:

Const add ={Strrt:’’,city:’’};

Const{strrt: st }=add;

1. Spread operator:

Const first = [1,2,3];

Const second = [5,6,7];

Const combined = first.concat(second);

Const combined1 = […first,’a’,…second];

We can apply it to object and clone the object.

1. Class:

Blueprint and we can create number object so we avoid code duplicacy.

class Person{

constructor(name){

this.name=name;

}

Walk(){

Console.log(“inside walk”);

}

}

Const p = new Person(“test”);

1. Inheritance:

Class Teacher extends Person {

Constructor(name,degree){

Super(name);

This.degree = degree;

}

Teach(){

Console.log(‘teach’);

}

}

Const teacher = new Teacher();

1. Modules:

-split the code into multiple file called modulize.

By default class in private so we have to export it

Export Class Teacher extends Person

Import {Person} from ‘./person’;