Name : Mayur Purushvani

Arrow Function :

We can use arrow function I 3 ways :

- For single argument
- For two arguments
- For multiple arguments

Let’s look at the example :

```
//Arrow function
const years = [1997,1998,1999,2000,2001];

//ES5
var years5 = years.map(function (el) {
    return 2021 - el;
});
console.log(years5)

//ES6
let years6 = years.map(el => 2021-el);
console.log(years6);                    //For Single argument

years6 = years6.map((el, index) => `Age of element ${index + 1} : ${el}.`);
console.log(years6);                    //For Two arguments

years6 = years6.map((el2, index) => {
    const now = new Date().getFullYear();
    const age = now - el2;
    return `${ el2 }`;
})
console.log(years6);                    //For multiple arguments
```

<u>Arrow with this keyword :</u>

The scope of this keyword is in the block only.

So, In below example, you can see that I've not used this keyword in the parameters like this.position or this.color. I've declare this as a self, then I used self keyword in my code. Otherwise it will give me undefined.

This will not work in ES5.

```
//ES5
var box5 = {
   color:'green',
   position : 1,
   clickMe : function() {
      var self = this;
      document.querySelector('.green').addEventListener('click',function() {
         var str = 'This is box number ' + self.position + ' and it is ' + self.color;
         alert(str);
      });
   }
}
box5.clickMe();
```

The same code in ES6 is something like this :

```
//ES6

const box6 = {
   color:'red',
   position : 1,
   clickMe : function() {

      document.querySelector('.red').addEventListener('click', () =>{
         var str = `This is box number ${this.position} and it is ${this.color}`;
         alert(str);
      });
   }
}
box6.clickMe();
```

You can see the difference between with arrow function and without arrow function.

The arrow function determines the this keyword and works fine.

But in ES5 we have to declare this on the upper function or assign someone else and used that in your code. Like we did in ES5 example.

Another example :

```
//ES5
Person.prototype.myFriends = function (friends) {

   var arr = friends.map(function(el){
      return this.name + ' is friends with ' + el;
   }.bind(this));
   console.log(arr);

}
var friends = ['anjali','vanshika','akshay','vaishnavi','bhoomi'];
new Person('mayur').myFriends(friends);
```

I can also use ‘bind()’ method in the inner function. This trick is most relevant then others.

This code is in ES6 is like this :

```

//ES6
Person.prototype.myFriends6 = function (friends) {

   var arr = friends.map(el =>
      `${this.name} if friends with ${el}`);
   console.log(arr);
}
var friends = ['anjali','vanshika','akshay','vaishnavi','bhoomi'];
new Person('mayur').myFriends6(friends);
```

<u>Destructuring :</u>

The destructuring is used to reduce code. This is how the new features comes in javaScript ES6 :

Let’s take an examples :

```
//ES5
var mayur = ['mayur',22];
var name = mayur[0];
var age = mayur[1];
console.log ( name , age);


//ES6
const [name1, age1] = ['mayur', 22];
console.log(name1,age1);

const obj = {
    firstName  :  'mayur',
    age2  :  22
};

const {firstName,age2} = obj;
console.log(firstName,age2);

const {firstName : a, age2 : b} = obj
console.log(a,b);



function calcAgeRetirement(year)
{
    const age = new Date().getFullYear() - year;
    return [age, 65 - age];
}

const [age3,retirement] = calcAgeRetirement(1999);
console.log(age3);
console.log(retirement);
```

Above, We have used objects via it’s name like I did in above example.

And same in a function, to call some methods I’ve created one array arguments and pass that in the function name only.

Array :

In ES6 we have some new features to reduce the code and also for better execution :

Let’s look at the below example :

```
const boxes = document.querySelectorAll('.box');

//ES5
/*var box5 = Array.prototype.slice.call(boxes);

box5.forEach(function(cur) {
     cur.style.backgroundColor = 'red';
});
*/

//ES6
const box6 = Array.from(boxes);
box6.forEach(cur => cur.style.backgroundColor = 'green');



//ES5
/*for(var i = 0; i<box5.length; i++)
{
   if(box5[i].className === 'box blue')
   {
       continue;
       //break;
   }
   box5[i].textContent = 'I Changed to blue';
}*/


//ES6
for(const cur of box6)
{
   if(cur.className.includes('blue'))
   {
       continue;
   }
   cur.textContent = 'I changed to blue';
}


//ES5
var ages = [12,17,8,21,14,11];
var full = ages.map(function (cur) {
   return  cur >= 18;
});
```

```
console.log(full);

console.log(full.indexOf(true));
console.log(ages[full.indexOf(true)]);


//ES6
console.log(ages.findIndex(cur => cur >= 18));
console.log(ages.find(cur => cur >= 18 ));
```

In this code, we have some different examples, like we've seen the Array.prototype.slice.call() method in ES5 but in ES6, there is only Array.from(). That's it.

Then we've seen the foreach() loop for both the versions.

Then we've seen the for loop of both the versions.

And at last we've seen one array example, that in ES5 we used indexof() method and in ES6 we use findIndex() or find() only.