ES-6

In scoping we have var,let and const:

**var:**

if we use var it can used at any where in the program with same variable name. For example

d=()=>{

var c=true;

if(c===true)

{

var c=false;

console.log(c) //here it prints false

}

console.log(c) //here also it prints false because it changes permanently

}

d()

in this example we use c is variable used in different ways. So we call it as function scope.

**let:**

if we use let it should be work for block scope

d=()=>{

let c=true;

if(c===true)

{

let c=false;

console.log(c) //here it prints false

}

console.log(c) //here also it prints because because let is a block scope

}

d()

**Const:**

If we use const before any variable the value that can not be changed it should be same value throughout the function/script. For example:

const c="xyz";

c="abc";

Here it throws an arrow like invalid assignment to const `c'

Different Ways to use Arrow Functions:

Suppose we have take an example of regular function in ES-5 without any parameters:

var c=6

var x=function()

{

return c

}

console.log(x())

If we take this example in ES-6, it would be written as

let c=6;

let x=()=>c

console.log(x())

We can declare same function in another way

let c=6;

let x= \_ =>c

console.log(x())

We take above example with parameters:

var c=6;

c=(a,c)=>{return a\*c}

console.log(c(3,8))

In this example c is overwritten by new value .i.e, is 8

Suppose if we use any predefined function, we can use arrow function like this

let x=()=>

{

this.val=1

setTimeout( \_ =>{

this.val++;

console.log(val);

},1000)

}

x()