**let** has block scope where as **var** has global scope, If we declare any variable with var inside a block it can be accessed globally.  
Example:  
 let globalVar = "I am global";

if(true){

let localVar = "I am local";

console.log(globalVar+" "+localVar);

}

console.log(globalVar);  
OUTPUT:  
I am global I am local  
I am global  
  
If we try to access local Variable outside we get an error.  
  
 let globalVar = "I am global";

if(true){

let localVar = "I am local";

console.log(globalVar+" "+localVar);

}

console.log(localVar);

console.log(globalVar);

OUTPUT: VM499:6 Uncaught ReferenceError: localVar is not defined

at <anonymous>:6:13  
  
If we re-declare the variable as **var** then we can access the local

Variable.  
  
   
  
  
  
let globalVar = "I am global";

if(true){

var localVar = "I am local";

console.log(globalVar+" "+localVar);

}

console.log(localVar);

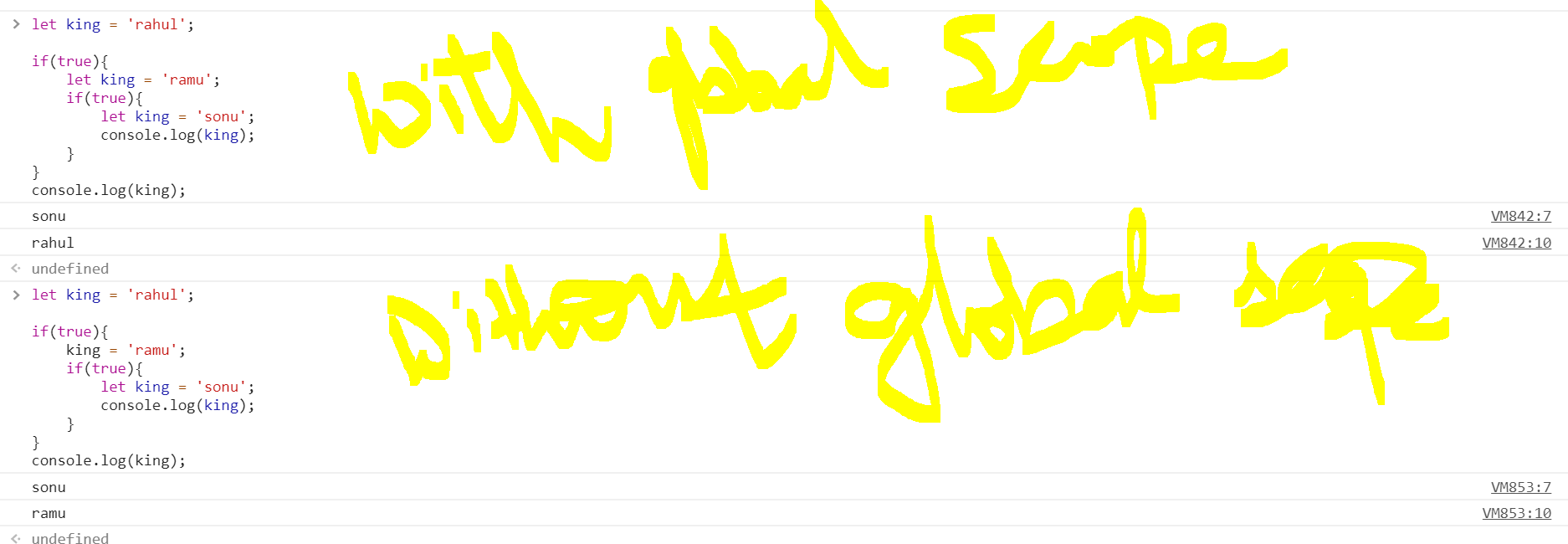
console.log(globalVar);

OUTPUT:  
 I am global I am local

I am local

I am global  
  
==============================================================================

Kings problem in JS  
  
1) When we won’t declare any datatype in JS by default it takes global scope.

  
  
Objects need to be separate by comma(,) and each object should have (: ) not equal.

Example:   
  
var Meeting = {

    var1:'Test',

    Time:6,

    var3:3,

    Schedule: function(room){

        console.log(`Your meeting is scheduled at ${this.Time} Clock at ${room}.`);

    }

};

In the above example we find that Variable time is being accessed using this, I we try to access it without **this** then we will get ERROR saying (Time is not defined). This is because we don’t have access to Time variable outside the scope. **this** allows the user to access the variables and methods which are outside the scope of the method.  
A sample application to schedule a meeting and reset the variables  
var Meeting = {

    var1:'Test',

    Time:6,

    var3:3,

    Schedule: function(room){

        console.log(`Your meeting is scheduled at ${this.Time} Clock at ${room}.`);

    },

    Reset: function(){

        this.var1 = '',

        this.var3 = 0,

        this.Time = 0

        this.print();

    },

    print: function(){

        console.log(this.var1+" "+this.var3+" "+this.Time);

    }

};

Meeting.Schedule('Zeal');

Documentation has everything just look and play.