18)variable scope: 1

let varOne = 'varOne';

if (true) {

console.log(varOne);

let varTwo = 'varTwo';

console.log(varTwo);

}

console.log(varTwo);

here we get error in last line.

Error is varTwo is not defined. But inside if block, we have are able to log that value.lets see scoping rules in js.

Javascript uses lexical scoping, it is sometimes also called static scoping. It is idea that variable defined in one part of your program might not be accessible everywhere else in your program. That is exactly what we are seeing in above code. The thing we want to pay attention to when it comes to scoping is code blocks in our script. So the only thing in above code that creates a code block is if statement. There are other structures in language that will also create code blocks but we will get to those later.

So when we used {} in if statement we created code block.

{

let i =1;

}

console.log(i);

output – i is not defined

Now what role code bocks play in scope? There are 2 types of scopes. One is global scope and other is local scope. Global scope contains all of things defined outside of all code blocks. In our code we only have 1 code block. Here varOne is defined outside of it so it is in global scope. So it is global variable.

Local scopes are things defined inside local scope. In our code we have one code block. So varTwo is in local block. So varTwo is local variable. This creates a scope hierarchy. In a scope you can access variable that are in that scope or in any parent/ancestor scope. Lets make a scope try. Now global scope is at root, it has no parent. So in global scope we can only access variable of global scope.

In local scope we can access variable of that scope and variable that in any parent/ancestor scope. Now here we do have a parent scope, it is global scope.

This explain the reason why varTwo was not accessible in global scope.

Code-

let varOne = 'varOne';

if (true) {

console.log(varOne);

let varTwo = 'varTwo';

console.log(varTwo);

if (true) {

let varFour = 'varFour';

}

}

if (true) {

let varThree = 'varThree';

}

console.log(varTwo);

here we have created a new independent scope and one nested scope.so scope of varFour is nested one. So inside scope of varFour we can access variable of that scope and variable of parent scope. Now this cope has 2 parents, global scope and scope of outer if.

19)variable scope: part 2

We can define variable with same name in different scope. They dnt interfere with each other. but we cannot define more than once in scope. Code-

let name = 'Sumit'

if (true) {

console.log(name);

if (true) {

let name = 'Mike';

console.log(name);

}

}

if (true) {

}

Now first consol.log, prints sumet second one prints mike. this is called variable shadowing

if (true) {

if (true) {

name = 'Mike';

console.log(name);

}

}

if (true) {

console.log(name);

}

In this case we see same value in both cases. It is because when we dnt declare a variable and we do write operation, then a variable is declared in global scope. This is called leaked variable. How to avoid? Always declare your variables. So let bounds variable with a scope.