**Chapter 7: Scope**

Scope is simply the area enclosed by {} brackets.

They are of three types:

* Global scope
* Block scope
* Function scope

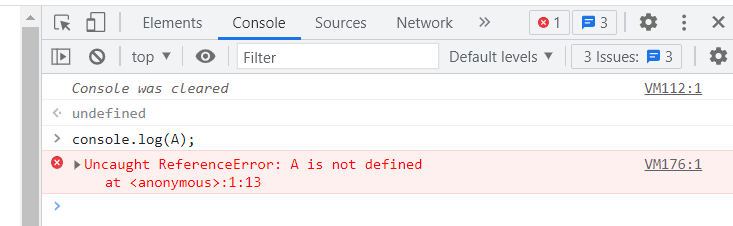
**Variable Definitions:**

Variables are case-sensitive.

Let a=1;

Let A=”apple”; both the variables are different.

Variables can be declared using var, let, const.



If a variable is referred without declaring it will throw an reference error.

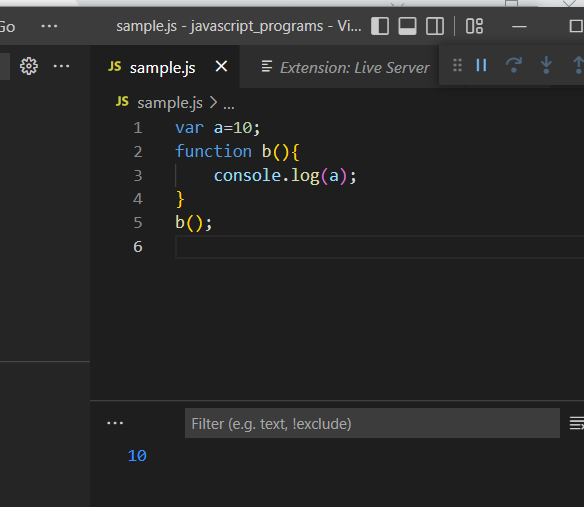
Var apple=1;

{

console.log(apple);

}

Here apple is defined in global scope. So it can be accessed anywhere in the program.



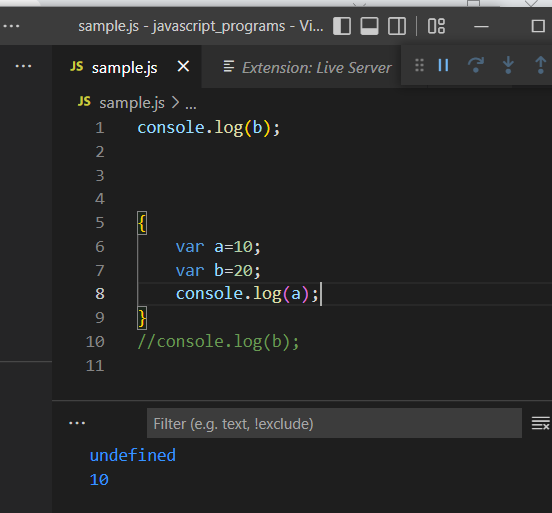
Here a is declared as var and is assigned a value 10 in the global scope. This variable can be accessed in anywhere in the program.

**Hoisting:**

Hoisting refers to the process of moving the declaration of functions, variables or classes to the top of their scope, prior to execution of the code.

Hoisting is limited to variables defines using var keyword and function name defined using function keyword.

Variables defined using let and const are not hoisted and their use remains limited only to the scope in which they were defined.

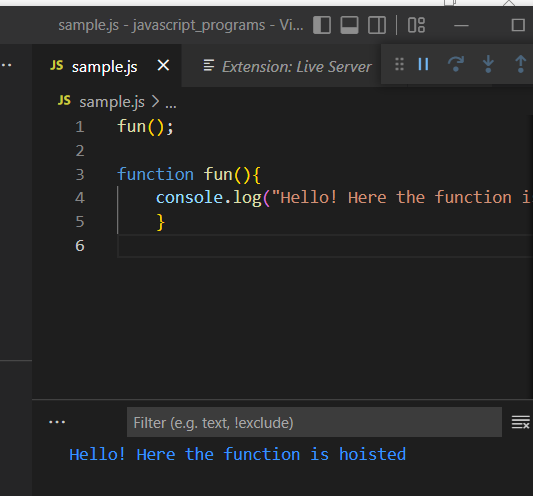


In the above example variable a and b are declared as var and are declared inside a block. When the variable b is accessed it displays undefined because it is declared somewhere in the program and the value is not assigned to it. The value of a is displayed as 10.

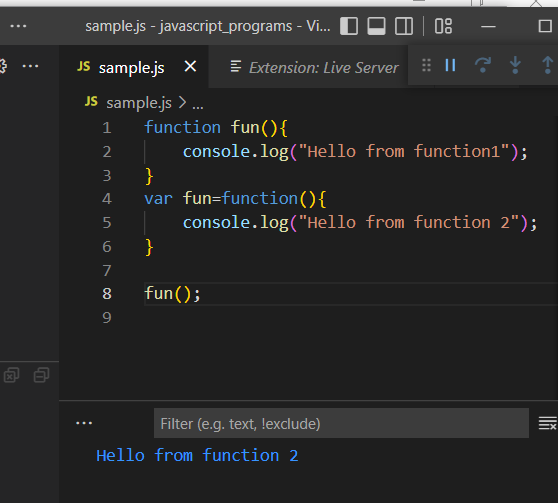
When a variable is defined in global scope it will propagate to every other scopes defined in the global context, including bock-level scope, for-loop scope, function-level scope, and event callback functions created using setTimeout, setInterval, or addEventListener functions.

**Function Name Hoisting:**

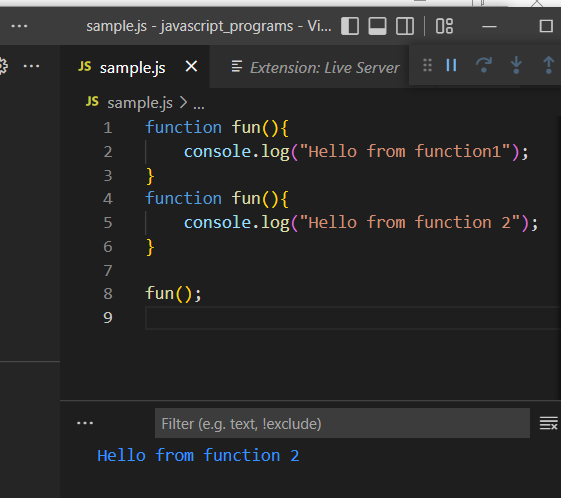
A function can be called anywhere in the program even before its definition is given.



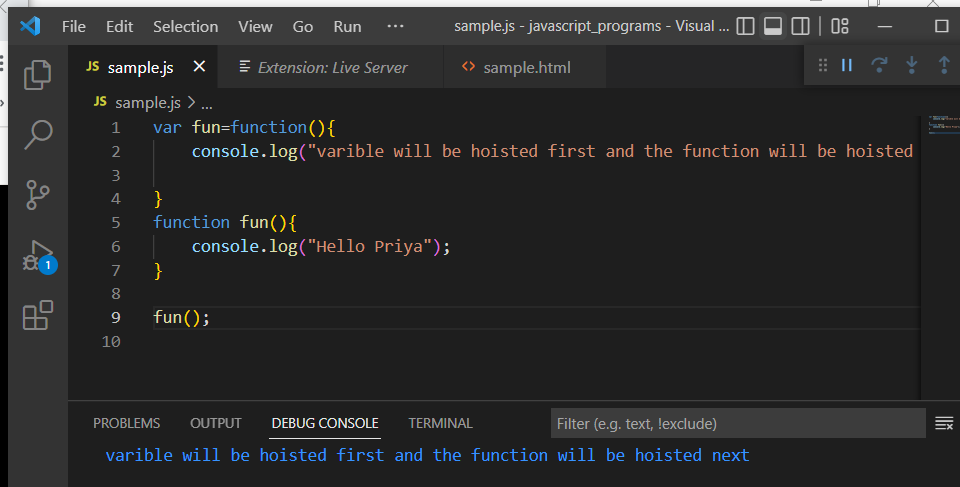
Here the fun() function is hoisted.



In the above example there is a function with the name fun() and a new variable fun is created and assign danother function to it, and when it is called it will not display any error rather the last defined function will be hoisted in javascript.

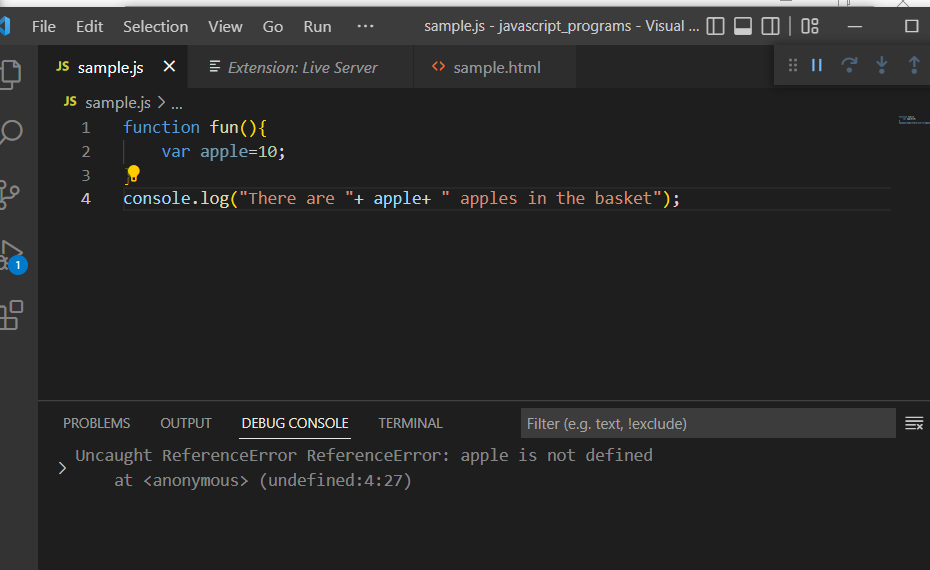


When a fun() is defined twice with the same name and it is called then the last defined function will be hoisted.



In case of variable and a function, both having the same name and when it is called the last defined function will not be called instead the variable will take the precedence over the function.

**Defining variables inside function scope:**



A variable declared inside inside the function will have its limitation in the function only.

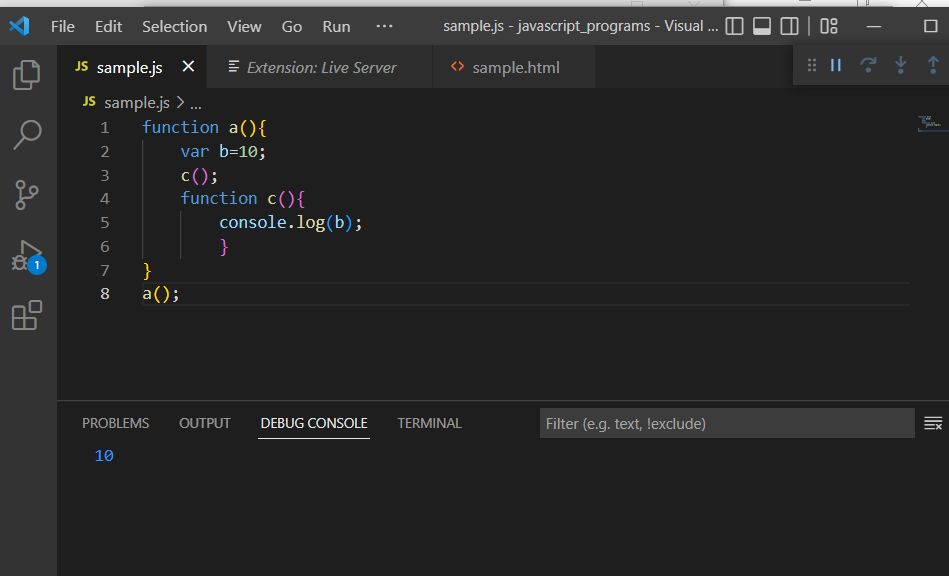
**Variable Types:**

* Var
* Let - its limit inside the scope it is defined
* Const - it is similar to let but the value cannot be reassigned.

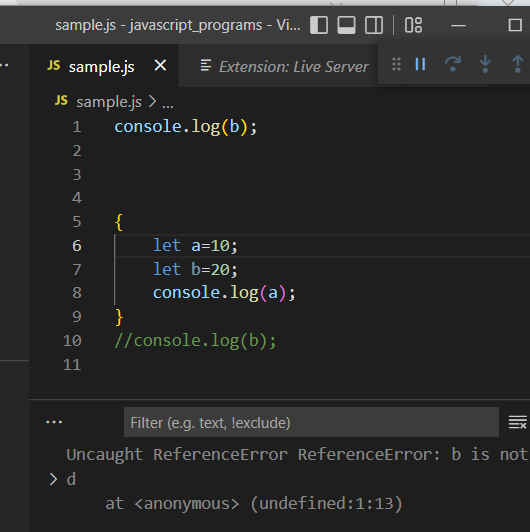
**Scope visibility difference:**

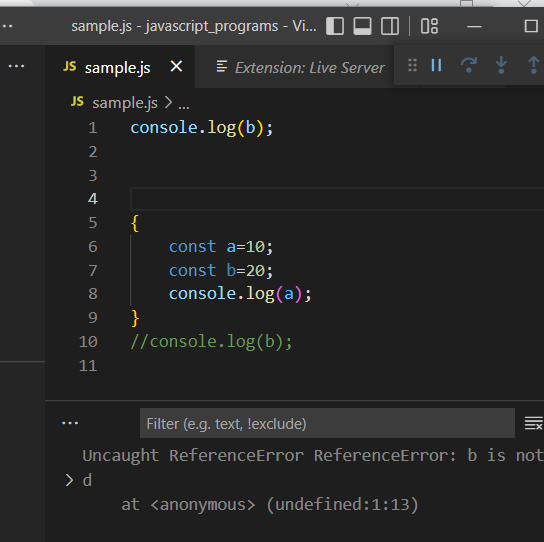
All variable types behave the same in the global scope. It can be accessed anywhere in the program.

Closures:



Here the function c () is wrapped inside another function a(). This is called closure. When the function a() is called , it moves to a() and in that function var iable b is defined and the function c() in called and as well as defined. Since c() is inside a() the scope of b also applies to c(). Thus the value is accessed and displayed in console.





When a variable is declared as let and const it cannot be accessed outside the block. Only var type has the access outside the block. But all the types are limited when it is defined inside a function.all the three types doesnot have scope outside the function.

**Const:**

When a variable is declared as const its value cannot be changed in any part of the program. Whenever a variable is declared const it should be defined else it will throw an error.

**Const and arrays:**

Changing a value in a const array is still allowed.

Const A=[];

a[0]=”a”; // allowed

A=[]; // this will throw an error. We cannot assign to the constant variable.

**Const and object literals:**

When an object is declared as const it makes the definition constant and the values of the properties can be changed.

