**作用域**：

是代码定义标识符(变量，函数名)的区域。

它是一套既定的规则，规定了如何查找变量，也就是确定当前执行代码对变量的访问权限。

JavaScript使用的是词法作用域(也称静态作用域)。

(是一种规则，它规定了如何去查找变量，比如说js是词法作用域，作用域在定义的时候就决定了)

词法作用域：函数的作用域在函数定义的时候就决定了。Why？答案在注释1，注释2

动态作用域：函数的作用域在函数调用的时候就决定了。

**作用域链：**

是一条有顺序的变量访问路径。

是由当前环境与上层环境的一系列**变量对象**组成，它保证了当前执行环境对符合访问权限的变量和函数的有序访问。

当查找变量的时候，会先从当前上下文的变量对象中查找，如果没有找到，就会从父级(词法层面上的父级)执行上下文的变量对象中查找，一直找到全局上下文的变量对象，也就是全局对象。这样由多个执行上下文的变量对象构成的链表就叫做作用域链。

执行上下文函数入栈时：

当函数调用入栈时，在创建阶段，会创建变量对象，建立作用域链，确定 this 指向。

怎么保存作用域链的？过程：注1 注2

函数有一个内部属性 [[scope]]，当全局代码执行，创建变量对象，函数被创建的时候，就会保存所有父变量对象到其中，你可以理解 [[scope]] 就是所有父变量对象的层级链，但是注意：[[scope]] 并不代表完整的作用域链！

当函数进入执行上下文栈，进入创建阶段时。会复制 [[scope]] 属性来创建作用域链。当创建 VO/AO 后将其放入作用域链顶端。这样就构建了完整的作用域链

var a = 10

function test() {  
 var a = 20

return a  
 }

test()

上述代码的作用域链(scopeChain)：[ AO(test), VO(global) ]

Let const 声明的变量的作用域

*for* (let i = 0; i < 3; i++) {

setTimeout(() => {

console.log(i) *// scopes: Local Block Global*

}, 10)

}

*// let const 声明的变量在作用域链上的形式*

*// function a() {*

*// let b = 'b'*

*// var c = 'c'*

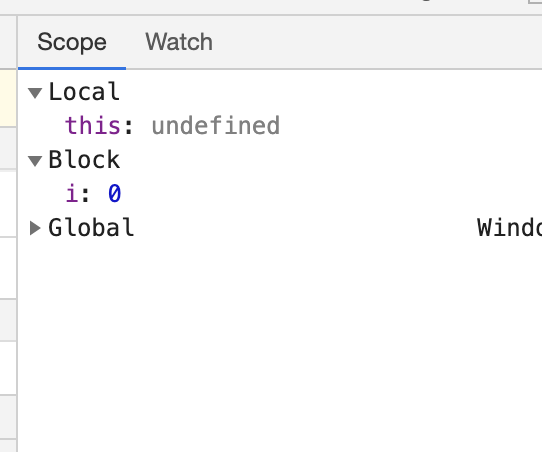
*// setTimeout(() => {*

*// console.log(b, c)// scopes Local Closure Global*

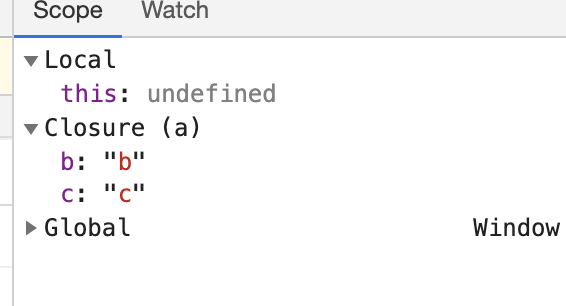
*// })*

*// }*

Let const



var



//全局

