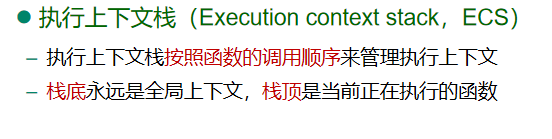
**执行上下文：函数每调用一次。就会创建一个新的执行上下文。函数执行结束，相应的执行上下文销毁。**



**调用栈：**

**以栈的方式来处理和追踪函数调用。**

**占地对应的是全局上下文环境。而站定对应的是当前正在执行的上下文环境。**

**return语句：**

**返回值**

**终止函数的执行。<===销毁当前执行上下文，弹出调用栈**

**Var a=20;**

**Function add(){**

**Var B=30;**

**var c=a+B;**

**Return c;**

**Var sum=100;**

**Console.log(sum);**

**}**

**由于return语句有终止函数执行，因此，return c之后的操作是无法进行的。**

**Var a=20;**

**Function add(){**

**Var B=30;**

**var c=a+B;**

**Return c;**

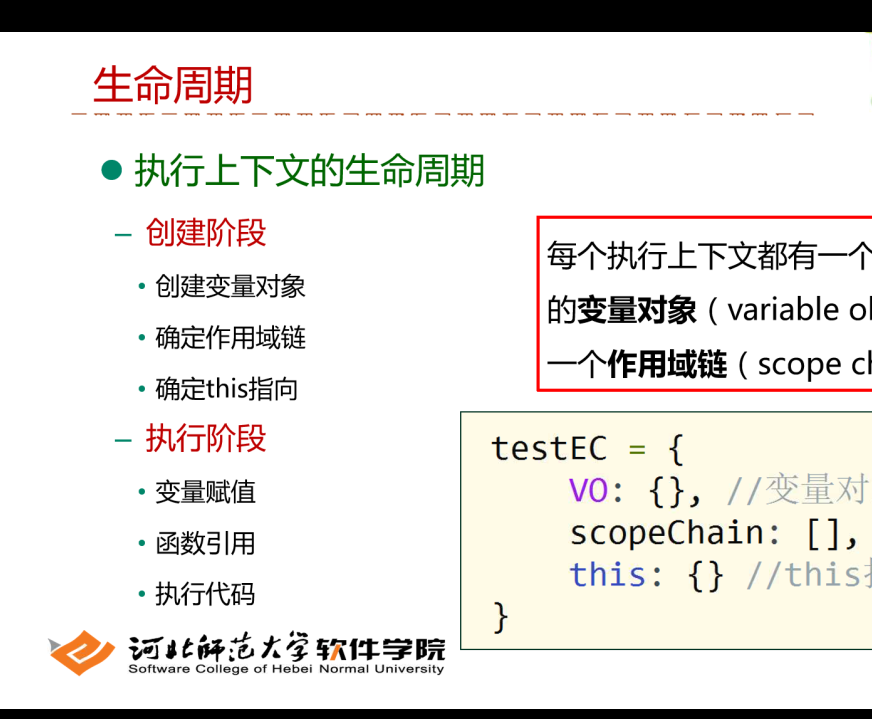
**Function bba(){**

**Console.log(B);**

**}**

**}**

**这个函数add里的函数bba是可以执行的，因为声明提升中的函数提升，把函数bba整体提到了var B=30等前面，因此，这个函数可以执行。原因请往下看。**



**创建变量对象：包含形参和实参**。

**什么是变量对象？？**

函数的所有形参（局部变量）；当前上下文中的所有函数声明；当前上下文中的所有变量声明。

**用var 声明的都是变量声明。**

**预编译**

Javascrict引擎是一段一段运行代码的。<---以函数为单位

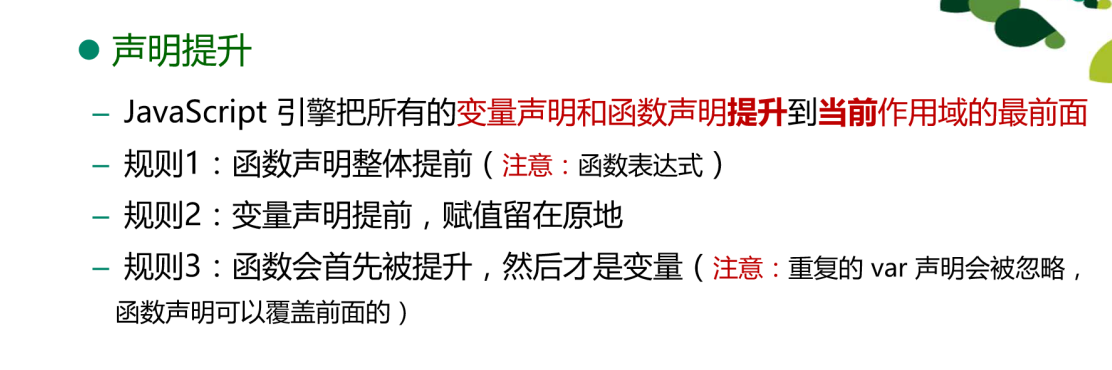
执行每一段代码之前都有一个“预编译”。

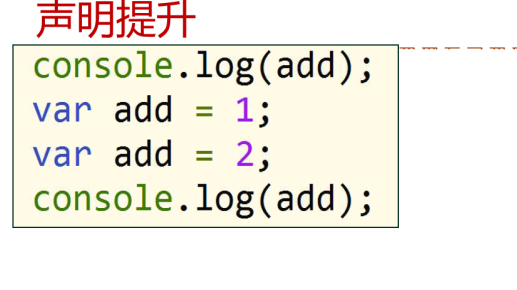
与C语言不同，C语言是全部编译再全部执行。

预编译的任务是：声明提升。

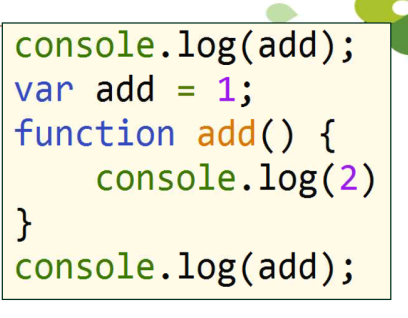
（作用域有两个，一个是全局作用域，一个是函数作用域。函数作用域可以访问全局作用域）

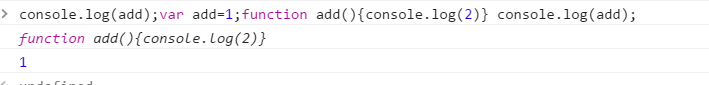
**声明提升：**



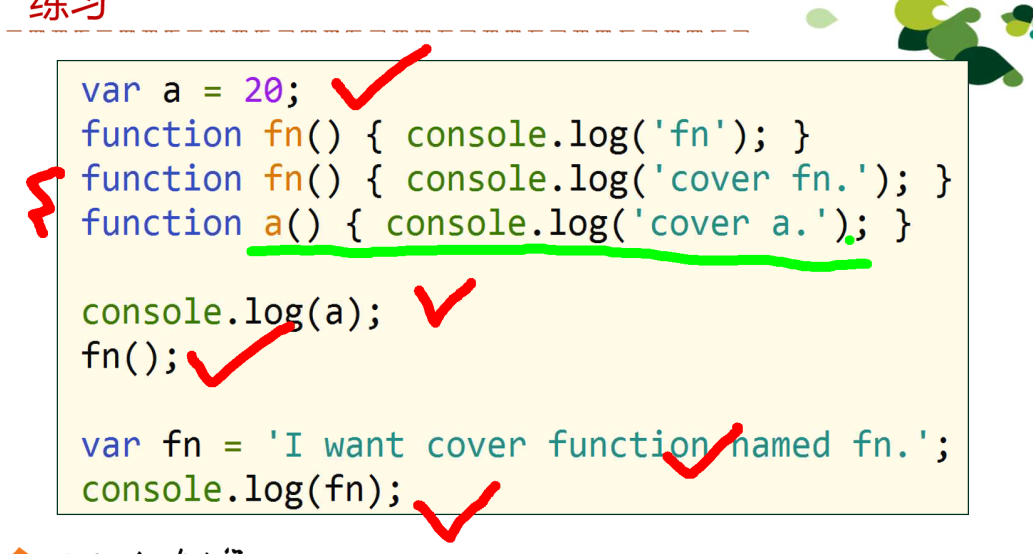


**看规则2，变量声明提前，赋值留在原地。**

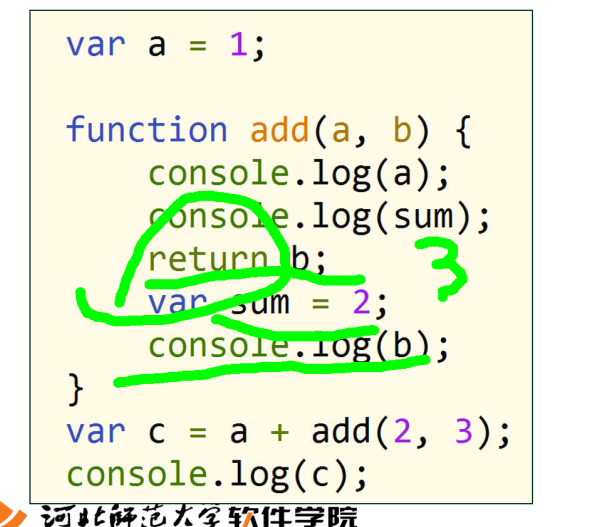




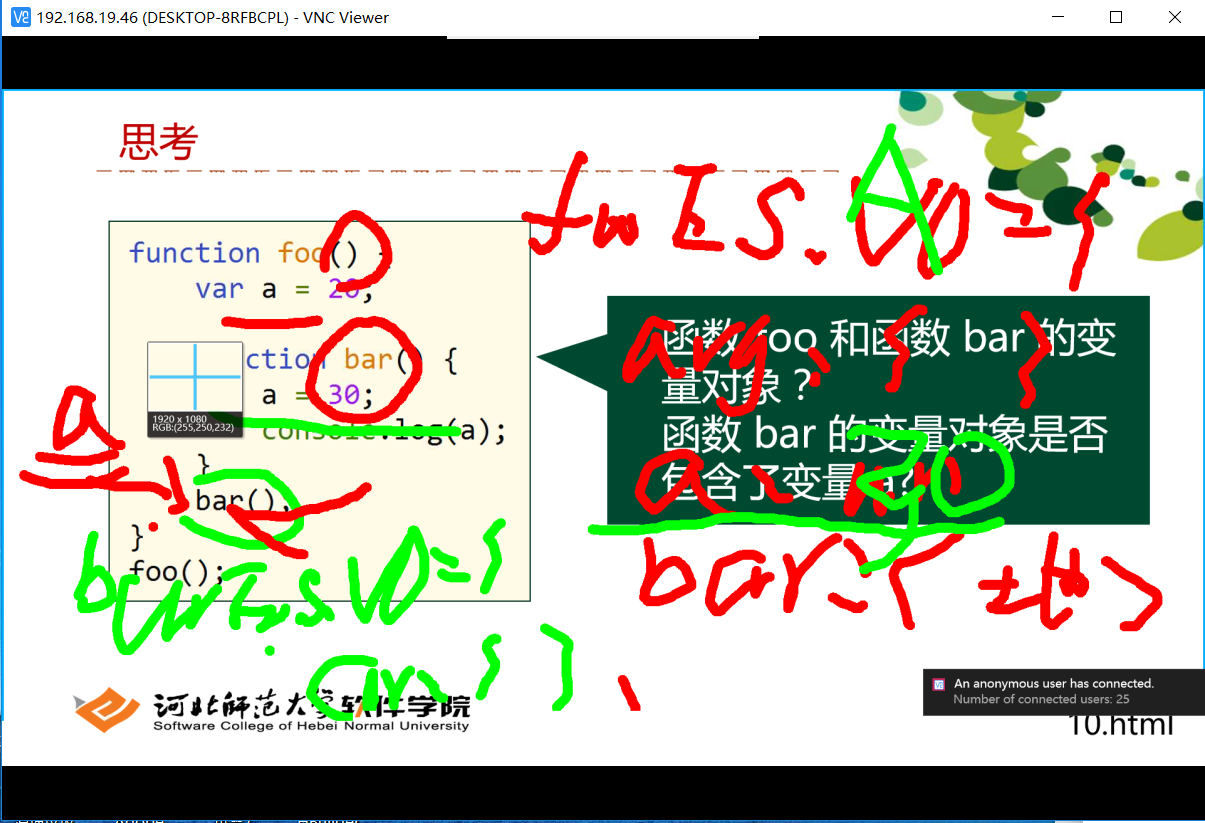
先提升函数function add；再走到var add=1;时因为跟函数add名字相同，所以被忽视了，先走console.log(add)，然后输出函数add；之后再走var add=1；然后走console.log(add)，输出2。



a被重新赋值了，函数a声明提升之后，再栈区的具体值是一个地址，堆内存开辟了一个空间，这个地址指向了这个空间。然后按顺序走的时候，先走了函数fu,函数a，然后到var a=20，所以a被重新赋值了，函数a中具体值被赋值成为20，所以console.log(a)输出的是20。



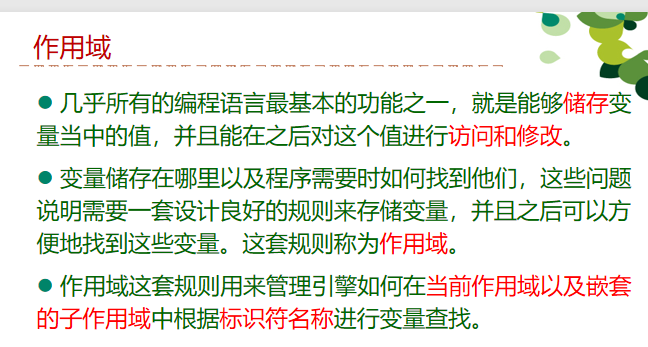
输出b之后，var sum=2;console.log(b);这两条语句就不会运行，因为return b运行之后就结束了。



在bar();后面加上console.log(a)；会输出30。因为在执行function bar(){}之后，a的值已经从a=20该成了a=30.

变量对象有三类：形参，函数声明，变量。

**作用域**

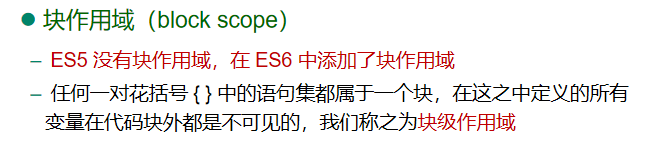


**什么是作用域：**

**作用域就是变量与函数的可访问范围。**

**作用域控制着变量与函数的可见性和生命周期。**

**作用域分为三类，分别为：全局作用域，函数作用域，块作用域。**

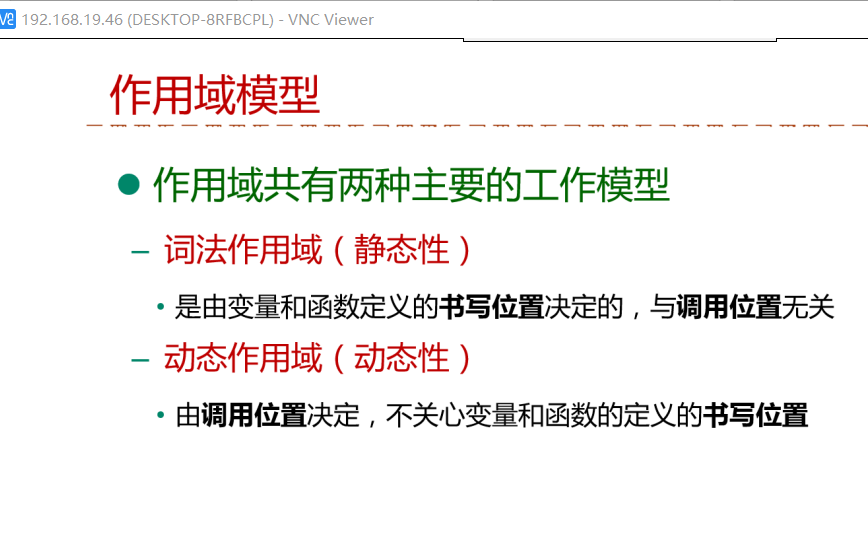


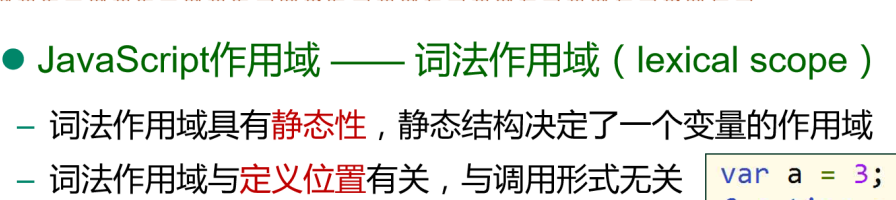
**全局变量：**（全局作用域）声明在函数外部的变量（所有没有var直接赋值的变量都属于全局变量）

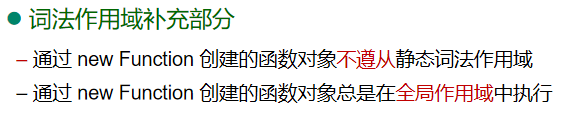
**在函数中使用var关键字进行显式申明的变量是做为局部变量，而没有用var关键字，使用直接赋值方式声明的是全局变量。**

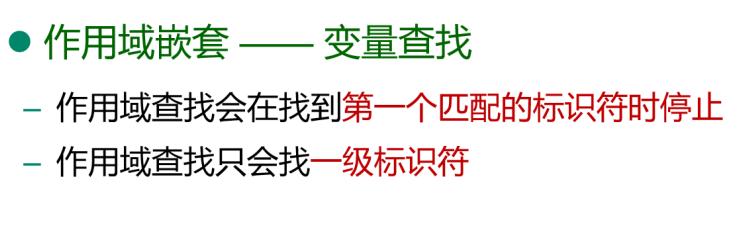
**全局变量在整个上下[文都](https://www.baidu.com/s?wd=%E6%96%87%E9%83%BD&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://blog.csdn.net/limu693992297/article/details/_blank)有效只是在没有赋值之前调用，会输出undefined**

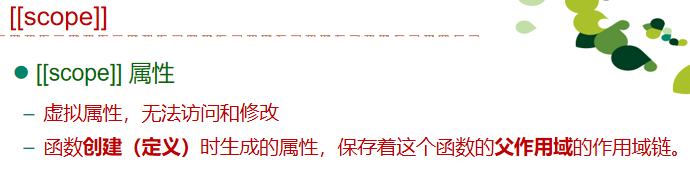
**局部变量：**（函数作用域）声明在函数内部的变量（所有没有var直接赋值的变量都属于全局变量）











**作用域链：**

**有当时执行环境与上层（父级）执行韩静的一系列变量对象组成。**

**提供对变量和函数访问的权限和顺序的规则。**