# 

**7**

new new Foo().getName()

=> let f = new Foo()

=>f.getName() = >AF3

=>new AF3()

Let f = new Foo()

形参赋值 变量提升

创建一个实例对象，this指向实例对象

代码执行

=>3

f.getName()

=>AF3()

形参赋值 变量提升

代码执行

=>3

**6**

…

f实例

实例对象

let f = new foo();

形参赋值 变量提升

创建一个实例对象，this指向实例对象

代码执行

getName = function(){}

return 实例

new Foo().getName()

=>实例.getName()

Return 实例对象

形参赋值 变量提升

创建一个实例对象，this指向实例对象

代码执行

=>2

**5**

实例对象

…

new Foo.getName()

=>new AF2()

**4**

=>1

形参赋值 变量提升

代码执行

getName()

=>BF1()

不销毁

执行完销毁

**3**

=>1

Window.getName()执行=》BF1()

形参赋值 变量提升

代码执行

“console.log(1)”

return this；=》return window

Foo AF0

getName AF1

AF4

BF1

BF1

全局变量

getName = function…

形参赋值 变量提升

代码执行

AF0()普通函数执行

Foo().getName()

=>**1、**AF0执行**2、**getName()

执行完销毁

getName()

=>AF4()

**2**

形参赋值 变量提升

代码执行

=>4

getName()

=>AF4()

Foo.getName()

=> AF2( )

执行完销毁

形参赋值 变量提升

代码执行

=>2

**1**

“console.log( 4 )”

AF4

“console.log( 3 )”

“console.log( 2)”

“console.log( 5)”

AF3

getName: **AF3**

Function:prototype:\_\_proto\_\_

getName

“getName= function(){

console.log(1)}

return this;”

\_\_proto\_\_:Object.prototype

变量提升

代码执行

function Foo( ){ … };

Foo.getName = function

Foo.prototype.getName=function…

var getName = function( ){ … }

function getName( ){ … }

AF2

Prototype:对象

\_\_proto\_\_:Function原型

constructor：Foo

Foo的prototype

AF1

Prototype

Foo的堆（AF0）

全局作用域

变量存储区 值存储区

全局作用域