1. 函数有定义--调用--执行--结束几个阶段
2. 执行阶段执行上下文创建阶段：1.）生成变量对象(在执行代码之前就已经创建了对象与变量)

3）建立作用域链

3）定this指向

1. 声明提升规则：
2. 函数声明整体提前 2.函数声明提前，赋值留在原地 3.函数会首先被提升，然后是变量 4.函数声明有冲突，会覆盖；变量声明有冲突，会忽略
3. 作用域链：VO（自身执行的内容）+scope（由父级变量对象构成）
4. 垃圾回收：变量对象当不被任何引用时，就会回收（即内存不与任何关联），而执行上下文销毁。
5. 闭包由函数及创建该函数的词法环境组合成。这个环境包含了该闭包创建时所需访问的全部局部变量。
6. this是在函数运行期间绑定，与函数定义无关
7. this是被自动定义在所有函数的作用域中的一个关键字
8. 函数调用形式：1.作为函数直接调用（定时器函数，立即执行函数）（默认绑定） 普通函数形式调用，指向window对象2.作为对象方法调用（隐式绑定）
9. 作为构造函数调用（new关键字调用），指向实例对象（new绑定）4.通过call/apply间接调用（显示绑定）

10.this绑定的规则：当函数嵌套时，由于this无法继承传递作用域绑定，this指向window对象，所以用\_this,that,self锁定this，或用bind（）函数锁定。

11.函数作为对象的方法调用时，函数上下文指向这个对象。

var a = 100;

var obj = {

a: 300,

fun: function() {

console.log(this.a);

}

}

obj.fun(); //this指向obj，结果为300

var fun2 = obj.fun(); //this指向window，因为this的指向与调用形式也有关系

fun2; //结果为100

1. 数组中存放函数，被数组索引调用，this上下文指向这个数组

function fun() {

console.log(this.length);

}

var length = 10;

var arr = [100, 200, fun];

fun(); //10

arr[2](); //3,被数组索引调用，this上下文指向这个数组,所以this。length是数组的长度

1. 绑定的优先级：new绑定>显示>隐式(通过obj或者数组的形式调用)>默认

var obj1 = {

name: "tom"

}

function fun1(name) {

this.name = name;

console.log(this.name);

}

var old = fun1.bind(obj1); //通过bind锁定，使this指向obj1,所以old为tom

old = new fun1("lily"); //绑定的优先级：new绑定>显示>隐式(通过obj或者数组的形式调用)>默认

old; //输出结果为lily

1. 严格模式：把 “use strict；”放在函数体所有的函数语句之前
2. 普通模式下，函数体中如果一个变量没有声明就赋值，默认是全局变量，但在严格模式中，没有声明不可使用。
3. 在严格模式下，全局作用域中函数中的this指向的是undefined
4. 纯函数：相同输入得到相同输出,不会产生副作用
5. 高阶函数：对其他函数进行操作的函数。1.函数的参数为函数 2.函数的返回值为函数
6. 回调函数：一个函数被作为参数传递给另一个函数，并不会马上被执行，回调函数会在包含它的函数内的某个特定的时间点被回调。