# 闭包

## 闭包的概念：

**“官方”的解释是**：所谓“闭包”，指的是一个拥有许多变量和绑定了这些变量的环境的表达式（通常是一个函数）， 因而这些变量也是该表达式的一部分。

## 变量声明，函数声明

**Javascript代码在执行的时候，分两个阶段**

1、解析阶段 （声明提前）

2、执行阶段 （执行各种代码）

Alert(a)

Var a =1;

Alert(a)

a()

Function a(){

console.log(a)

}

a()

## 变量作用域

javascript没有块级作用域”。所谓“块”，就是大括号“｛｝”中间的语句。例如if、for语句：

**var** i = 10;  
 **if**(i>1){  
 **var** name = "asd";  
 }  
 console.log(name);//asd

javascript除了全局作用域之外，只有函数可以创建的作用域。

Function fn1(){

var a =2

}

Alert(a)

## 自执行函数

**function** (){})();

可以利用自执行函数创建作用域，而且执行完内存会立即回收；

var a = 2;

(**function** (){

var a =1;

alert(a)  
})();

alert(a)

## 未执行的函数

**函数声明和函数表达式**

function fn(){}

fn2 = function(){}

**未执行的函数，在被创建的时候做了什么呢？**

**var** x= 1;  
**function** fn(){  
 console.log(x)  
};  
  
**function** show(){  
 **var** x = 2;  
 fn();  
}  
show();

## 做了什么…

简单来说，定义函数的时候，实际上就是创建了一个对象，  
就这么简单！不过为了保证这个对象能够调用，它有几个特殊的内部属性/方法：  
[Call]  
[[Construct]]  
[[Scope]]  
[[Code]]  
[[FormalParameters]]

当调用函数的时候，实际上就是调用该函数对象的[[Call]]内部方法；  
当使用 new 调用函数的时候，实际上就是调用该函数对象的 [[Construct]] 内部方法。  
  
定义函数时，**{}** 之间的东西，不管是什么，哪怕是空的，全部原封不动地保存在 **[[Code]]** 内部属性里；  
  
将函数能访问到的"变量对象",绑定在它的**[[scope]]**上(用于变量查询);

**—————静态作用域链**

## 作用域链：

作用域链的**非自己部分**在函数对象被建立（函数声明、函数表达式）的时候建立，而不需要等到执行.

作用域链的前面部分是静态的，当**函数执行时**，建立一个自己当次执行的作用域，然后把这个作用域与前面共享的链关联起来;

## “闭包”

**>“官方”的解释是**：所谓“闭包”，指的是一个拥有许多变量和绑定了这些变量的环境的表达式（通常是一个函数）， 因而这些变量也是该表达式的一部分。 ————（太他妈抽象了）

**犀牛的解释：**函数对象可以通过作用域链互相关联起来，函数内部的变量都可以保存在函数作用域内，这种特性在计算机科学文献中称为“闭包”。

**从技术角度讲，javascript函数都是闭包，它们可以把父级作用域里面的变量保存在自己内部，包裹起来；**

**简单的理解：**闭包是指在函数**外部访问**函数作用域中变量的**函数（比较狭义，一种特殊情况）**

function fn1 (){  
 var a = 2;  
 return function (){  
 console.log(a)  
 }  
 }  
 var s = fn1();  
 var a =3;  
 s()

## 闭包的作用

**优势：**

1、加强了封装性

2、保存变量——在内存中维持一个变量(用的太多就变成了缺点，占内存)

**负作用：**

浪费内存

## 闭包的应用

1、保存变量。

(一个ul里面有5个li，点击某个li是，alert出它所在的位置)

2、实现面向对象的封装。

**function** person(name){  
 **var** name = name || “default";  
 **return** {  
 sayName : **function**(){  
 alert(name) ;  
 },  
 setName : **function**(newName){  
 name = newName;  
 }  
 }  
 };

## 重点知识点

1、变量作用域

2、自执行函数

3、没有执行的函数，在创建的时候干了什么！！！

4、变量的作用域链

5、闭包。

**1、什么是闭包？**

比较常见的闭包是指在函数外部可以访问函数内部变量的函数，因为一般情况下在函数外部是没法访问函数内部的变量的。比如说定义一个函数fn1，如果在函数fn1的内部再定义一个函数fn2，fn2是能访问到fn1的变量的，然后再把fn2 return出去，fn1外部就可以通过fn2访问fn1的变量了；

**function fn1（）{**

**var a = 1**

**function fn2(){**

**alert(a)**

**}**

**return fn2**

**}**

**var fn = fn1()//fn=>fn2**

**fn()//alert(1)**

**2、为什么fn2是能访问到fn1的变量的？**

因为函数被创建的时候，就把它能访问到的变量绑定在他的作用域链里面了，函数调用的时候会直接在作用域链里面找

闭包就是能够读取其他函数内部变量的函数。

由于在Javascript语言中，只有函数内部的子函数才能读取局部变量，因此可以把闭包简单理解成“定义在一个函数内部的函数”。