**栈的运用--调用堆栈**

由于javascript是一种单线程编程语言，这意味着它只有一个 call stack。Call stack中文意思就是【调用堆栈】。它记录了我们程序中执行的位置。如果我们执行一个函数，则是把它入栈，如果我们从一个函数返回，则是把它弹出堆栈顶部。看下列例子：

function b (x,y) {

Return x+y

}

function a(x) {

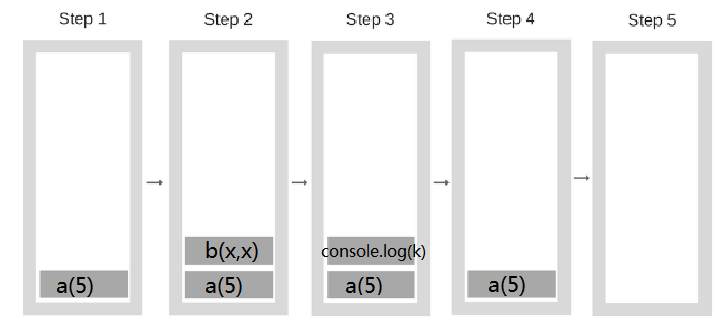
var k = b(x,x)

console.log（b）

}

a(5)

那么当引擎执行该段代码过程如下：



另外，当遇到抛错时，我们也是很容易看到栈结构的顺序的，比如以下代码：

function foo() {

throw new Error('SessionStack will help you resolve crashes :)');

}

function bar() {

foo();

}

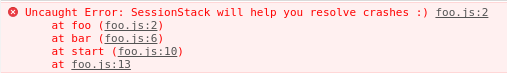
function start() {

bar();

}

start();

假设这段代码在foo.js文件中，并在chrome执行，则会出现以下错误：



**堆栈大小**

还有一种情况是调用堆栈情况达到最大值，比如以下代码：

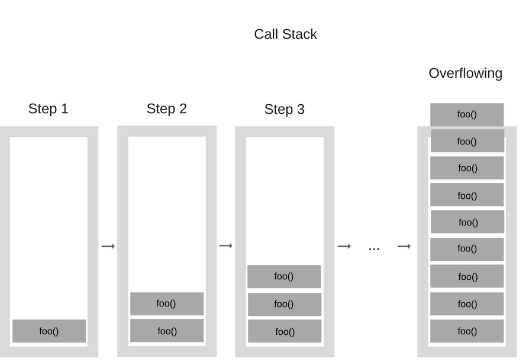
function foo() {

foo();

}

foo();

由于在引擎中执行这个代码时，会先调用foo函数，并且不断的递归调用，没有任何终止调整，所以执行时，同一个函数会不断重复的添加到调用栈中，如下图：



直到最后超出调用堆栈的实际大小，浏览器就会抛出如下错误：

IMG_256