day5-javascript函数下

笔记本： 凡尘：二阶段
创建时间： 2017/7/27 22:15 **更新时间：** 2017/9/28 20:29
作者： 437389128@qq.com

什么是作用域？

变量可访问的范围(起作用的范围)

全局变量&局部变量

0、函数的特点
1、每个函数都是一个独立的空间（小黑屋）
2、函数内部可以访问函数外部的变量和函数
3、函数外部不可访问函数内部的变量和函数
4、函数没有运行时，里面的变量是不存在的

1、全局变量
在任何地方都可以访问的变量

2、局部变量
局部：指的是函数内部

局部变量：在函数中使用var定义的变量、函数的参数、函数中的方法

作用域链（重）

1、什么是作用域链？（面试常问）

1、简单说就是作用域集合

2、当前作用域 -> 父级作用域 -> ... -> 全局作用域　形成的作用域链条

堆和栈

函数的值传递

堆栈是一个在计算机中经常用到的抽象数据类型，它是将数据分配到内存空间来完成各种调用(内存中除了有堆与栈还有一个常量池)

为什么内存里面有堆内存与栈内存之分？答：为了使程序运行时占用的内存最小

特性：最后一个放入堆栈中的物体总被最先拿出，这个特性通常称为后进先出队列

堆栈中最重要的是push和pop

a、push操作在堆栈的顶部加入一个元素。

b、push相反，在堆栈顶部移除一个元素，并将堆栈的大小减一

计算机删除文件为什么这么快？

因为文件没有删除。
删除的是索引文件。

计算机里，索引无处不在

变量不写var默认全局变量

```
fucntion test(){
    x = 999;
}
X是全局变量
```

函数的参数都是局部变量

```
var x = 100;
function test(x){
    x+=100;
}
test(x);
console.log(x);
```

变量作用域(闭包)

```
1.function outFn(){
      function innerFn(){
         var inner=0;
         inner++;
         console.log('inner='+inner)
      }
      return innerFn;
  }
  var fn1=outFn();
      fn1();
      fn1();
```

```
    //每当这个内部函数被调用的时候，都会重新生命一个inner变量，然后让这个变量自增。

2.
    var inner=0;
    function outFn(){
        function innerFn(){
            inner++;
            console.log('inner='+inner)
        }
        return innerFn;
    }
    var fn1=outFn();
        fn1();
        fn1();



//每当内部函数被调用时，都会让全局变量inner自增。

如果让这个变量变成父元素的局部变量，会是什么结果那？

function outFn(){

        var inner=0;
        function innerFn(){
            inner++;
            console.log('inner='+inner)
        }
        return innerFn;
    }
    var fn1=outFn();
        fn1();
    var fn1=outFn();
        fn1();
```

变量提升（重）

1、js代码的预编译和执行

2、变量的提升

```
<script type="text/javascript">
    /*
        预编译  执行

        预编译过程做的事情：

            把var 和 function定义的变量提升到script的最上方

            赋值语句是不会被提升的(哪怕=后面是一个function)

        变量提升

        注意：
            使用变量形式定义的函数只能在赋值语句之后调用

            test();
            // TypeError: test is not a function  test不是一个函数

            var test = function(){

            }

            test();
    */
```

变量的声明提升

```
tt='ee';
function  b(){
    function test(){
        alert(tt);
        var tt='ff';
    }
    test()
}
b()
```

```
var tt='123'
function test(){
      var tt ;
      alert(tt);
      tt='ff';
}
```

```
2、tt='ee';
function  b(){
   function test(){
        alert(tt);

   }
   test()
}
b()
```

实战

```
1、
     console.log(a);
     var a = 10;
     console.log(a);

2、
    console.log(test);
    var test = function(){
        console.log(1);
    }
    function test(){
        console.log(2);
    }
    console.log(test);
```

思考1

```
var age = 10;
function test(){
     console.log(age);
}
```

```
test();

打印什么？
```

思考2

```
var age = 10;
function test(){
    console.log(age);
    var age = 20;
    console.log(age);
}
test();

打印什么？
```

思考3

```
function test(){
    var age = 20;
    console.log(age);
}
test();
console.log(age);
打印什么？
```

思考4

```
var age = 10;
function test(){
    var age = 20;
    console.log(age);
}
test();
console.log(age);
打印什么？
```

思考5

```
var age = 10;
```

```
function test(){
    var age = 20;
    console.log(age);

    function inner(){
        var age = 30;
        console.log(age);
    }
    inner();
    console.log(age);
}
test();
console.log(age);
```

打印什么

console.log与console.dir的区别

`console.log()`可以取代`alert()`或`document.write()`，在网页脚本中使用`console.log()`时，会在浏览器控制台打印出信息。

`console.dir()`可以显示一个对象所有的属性和方法。

什么是构造函数

与类名称具有一样名称的成员函数是构造函数。构造函数不能有返回值,甚至不能有`return`语句

构造函数是干什么用的？

构造函数是初始化已创建好的对象中成员变量的

构造函数是什么？

构造函数创建后可以为它动态的添加属性和方法(两种方式)
当一个函数使用new 关键字调用时，则称为构造函数
var obj = new object();
typeof obj

。

var obj = {}

对象类型是什么？

是一堆信息打包后的产物，多种信息的压缩体。

对象类型的好处在哪？

使得信息的传递更方便快捷

练习：
创建一个对象，表示一个汽车的信息

```
function car(){
}
var car01 = new car();
car01.logo = "BENZ";
car01.speed = "180km/h";
car01.price = 450000;
car01.size = 5;
car01['size'] = 5;
```

创建一个对象，表示一个学生的信息

```
function student(){
}

var s1 = new student();
s1.name = "小明";
s1.age = 16;
s1.sleep = function(){
    console.log("zzzZZZZ....");
}
s1.eat = function(){
}
```

表示一个对象的行为时，可以使用函数来实现。

匿名函数（重）

1、什么是匿名函数？
没有名字的函数
function(){

}

2、匿名函数的作用
定义函数：
var fn = function(){

}
事件：
btn.onclick = function(){

}

匿名函数（重）

1、什么是匿名函数？
没有函数名的函数　function(){}

2、匿名函数有什么用？

a、当做值给某个变量赋值
var func = function(){}

b、事件驱动
div.onclick = function(){}

c、**当做参数进行传递(回调函数)**
arr.sort(function(){});

思考

定义一个加法功能，计算传入的所有数字的和，怎么做？

arguments（重）

1、函数内部自带的一个对象，形式和数组类似（被Arguments创造出来的）

2、存储的是所有的实参（调用功能传入的参数）

3、可以使用[]及下标访问arguments中的内容 **arguments[0]** 访问第一个实参

4、可以使用 **arguments.length** 确定传入实参的个数 （arguments['length']）

最常用的用途： 判断传入参数的个数(根据参数个数做不同的事情)

实战

1、定义一个加法功能，计算传入的所有数字的和

2、定义一个功能getMax，传入任意个数字，返回最大值

```
function getMax(){
    var max=0;
    for(var i=0;i<arguments.length;i++){
        if(max<arguments[i]){
            max=arguments[i]
        }
    }
    return max
}
```

3、定义一个运算功能，参数为(x,y),当传入1个数字时，求x的阶乘，如果传入两个数字，求x的y次方

```
function num(){
    var sum=1;
    if(arguments.length==1){
        for(var i=1;i<=arguments[0];i++){
          return  sum*= i;
        }
    }else{
        return Math.pow(arguments[0],arguments[1])
    }
}
```

美国作家罗伯特·海因莱恩的科幻小说《你们这些回魂尸》

什么是递归？

举一个简单的栗子

自己玩自己

```
function f(){
    console.log(1);
    f();
}
```

间接的玩自己

```
function f1(){
    f2();
}
function f2(){
    f1();
}
```

对于递归而言，最重要的是跳出结构，因为只有跳出结构才可以有结果

其实所谓的递归就是化归思想

递归调用函数，写一个递归函数，最终还是要转换为自己这个函数

递归的思想就是将一个问题转化为一个已解决的问题来实现。

例二：算100以内的和

```
//for循环来做
var sum=0;
for(var i=0;i<=100;i++){
    sum+=i
}
console.log(sum)

//while循环
var i=0;
var sum=0;
while(i<=100)
{
    sum+=i;
    i++
}
console.log(sum)
```

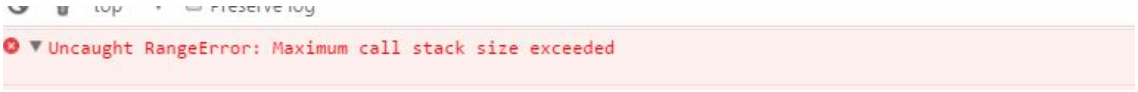
//下面我们用递归来做

1、首先假设我们已经写好了递归函数，假设就是fn(),fn(100)就是求1到100的和

2、寻找递归的关系，就是n与n-1，或者n-2之间的关系：foo(n)==n+foo(n-1);
第一次：100+foo(n-1)---->100+99
第二次：100+99+foo(n-1)---->100+99+98
第三次：100+99+98+foo(n-1)----->100+99+98+97

最后一次：100+99+98+97....foo(n-1)----->100+99+98+97+.....+1
1 2 3 4

```
function fn(n)
{
   return n+fn(n-1);
}
var sum=fn(100);//把函数执行返回后的结果返回给sum.想看当前函数的返回结果看函数里面是否有return
console.log(sum)
```



这个错误是什么？ 是内存溢出，也就是说因为不恰当的代码，导致了递归或者是死循环。递归错误超出了堆栈大小

```
function fn(n){
    if(n<1)
    {
        return 0;
    }
    return n+fn(n-1);
}
var sum=fn(100);
console.log(sum)
```

递归和循环的区别

递归与循环是两种不同的解决问题的典型思路。

递归算法：

优点：代码简洁、清晰，并且容易验证正确性。

缺点：
1、它的运行需要较多次数的函数调用，如果调用层数比较深，每次都要创建新的变量，需要增加额外的堆栈处理，会对执行效率有一定影响，占用过多的内存资源。

2、递归算法解题的运行效率较低。在递归调用的过程中系统为每一层的返回点、局部变量等开辟了栈来储存。递归次数过多容易造成栈溢出等

注意：递归就是在过程或函数里调用自身；使用递归策略时要注意的几个条件
1、必须有一个明确的递归结束条件，称为递归出口。
2、递归需要有边界条件、递归前进段和递归返回段。
3、当边界条件不满足时，递归前进。当边界条件满足时，递归返回。

循环算法：

优点：速度快，结构简单。

缺点：并不能解决所有的问题。有的问题适合使用递归而不是循环。如果使用循环并不困难的话，最好使用循环

练习：
递归计算100的阶乘

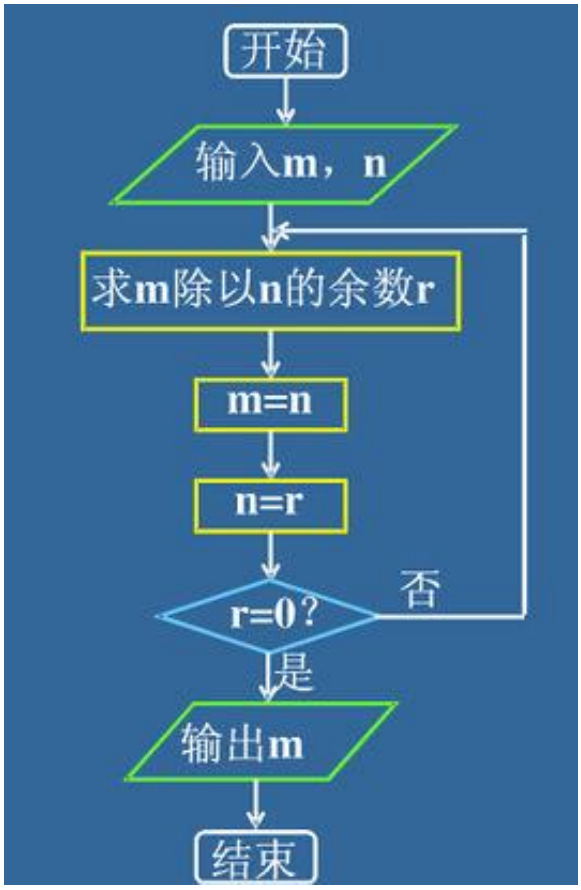
递归计算 斐波那契数列

假设有一对兔子，三个月成年，到了第三个月，就可以生下一对小兔子。
问，在第N个月，总共会有多少对兔子？

它后一个数等于前面两个数的和

1 1 2 3 5 8 13 21 34 55 89 144

递归计算两个数字的最大公约数



使用函数完成任意数字阶乘的计算
要求：页面输入任意数字，点击按钮后计算阶乘

综合练习：

预习

1、什么是数组？
2、数组的定义方式
3、数组常用api(pop/push/shift/unshift/sort　concat/join/splice/slice)
4、怎么遍历数组
5、什么是二维数组，什么是多维数组
6、排序算法（冒泡排序，选择排序，快速排序）