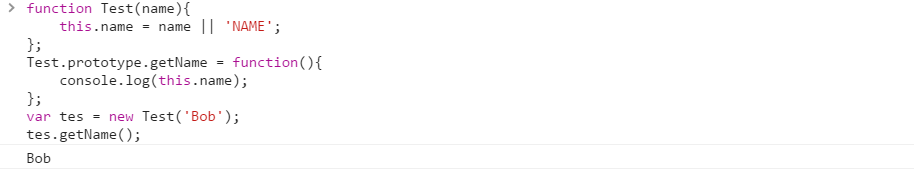
第二次更新

在函数中this的取值，实在函数执行的时候确定的。

再补充一点，就是this在原型链里面的问题，

就比如说，this在构造函数的原型prototype上的属性里的时候



此时this指向的还是new出来的实例对象。

第一次更新

以前每次遇到要用this都很头疼，老怕用不好，这次来一次系统的学习总结。

第一种情况：this在一个对象的属性里面调用。

这个时候this指向的就是本对象。

例子：

    var test1 = {

             x:0,

             y:0,

             fun:function(x,y){

                  this.x = this.x+x;

                  this.y = this.y+y;

                  console.log('x='+x);

                  console.log('y='+y);

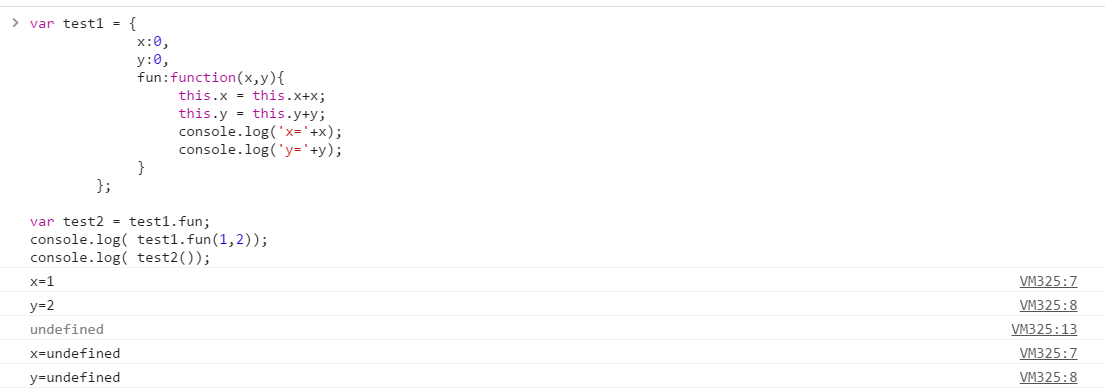
             }

        };

        test1.fun(1,2);  //x=1 y=2

再更新一点，我们上面是直接调用的这个对象的属性，this指向这个对象，没问题。

那么，我们如果说把对象的这个属性赋给一个变量，我们调用这个变量的话，this指向 的就是window了。下面看图



第二次输出的x和y都是undefind，此时this指向window说明window.x和window.y都没有赋值。

在这里做一点补充，我们再看个例子

其实就是函数套函数的情况下，

        var test = {

            x:0,

            y:0,

            fun:function(x,y){

                var test\_1 = function(x){

                this.x = x;  //这里的this其实绑定到了全局对象上

                }

                var test\_2 = function(y){

                    this.y = y;    //这里的this其实也绑定到了全局对象上

                }

                test\_1(x);

                test\_2(y);

            }

        }

        test.fun(9,10);

        console.log(test.x);  //0

        console.log(test.y);  //0

        console.log(x); //9

        console.log(y); //10

在这里，据说是js本身的设计缺陷，但是如果我们碰到这种情况该怎么办，很多厉害的前辈已经给出了方法，下面看例子

  var test = {

            x:0,

            y:0,

            fun:function(x,y){

                var that = this;  //就是先在this还指向本对象的时候，将其赋给that（一般我们默认为that）

                var test\_1 = function(x){

                    that.x = x;

                }

                var test\_2 = function(y){

                    that.y = y;

                }

                test\_1(x);

                test\_2(y);

            }

        }

        test.fun(9,10);

        console.log(test.x);  //9

        console.log(test.y);  //10

第二种情况：this直接在函数内调用，此时指向全局对象，但是一般不建议这样写。

例子：

function test2 (x) {

     this.x = x;

}

test2(2);

console.log(x);  //  此时x=2

第三种情况：this在构造函数里面（新手注意，调用构造函数的时候要用new调用才是真正的构造函数，否则和调用普通的函数一样）

这个时候，this指向的是new出来的实例对象，当然，如果直接调用构造函数，就和前面的情况一样了，this会指向全局对象window

例子：

function Test3(x,y){   //构造函数的函数名我们一般默认第一个字母大写

            this.x = x;

            this.y = y;

        }

        var test3 = new Test3(7,8);

        console.log(test3.x);

        console.log(test3.y);

看到这里，应该想起来一点java的面向对象编程的思想了。

第四种情况，通过apply或者call绑定,我的理解就是，改变this的指向

function Subway(name){

     this.name=name;

     this.speed=0;

     this.run=function(speed){

     this.speed=speed;

     };

 }

 var s=new Subway('1号线');

 s.run(300);

 console.log('一号线速度为：',s.speed);//300;this绑定到新生成的对象s

 var s1=new Subway('2号线');

 s.run.apply(s1,[100]);

 console.log('二号线速度为：',s1.speed);//100;this绑定到对象s1

 s.run.call(s1,200);

 console.log('二号线速度为：',s1.speed);//200;this绑定到对象s1

参考博客<http://www.cnblogs.com/greenteaone/p/4193985.html>

1