//想着写一个可以检测所有数据类型的函数

// by Herpang

先总结，可以检测数据类型的方法，和其优缺点

1,typeof、instanceof、 constructor、 prototype

先说typeof 用法typeop xxx 或者typeof() 返回值是一个string

只会返回number string boolean undefind object funtion 这几个值

基本数据类型中number string boolean undefind 会正常返回，

//而typeof null的话，会返回object， function 会正常返回function

对象，数组和null区分不了。

一个面试题 console.log(typeof typeof typeof function () {}); // string

//这里其实就是考察typeof的返回类型了

2,instanceof 这个方法是检测是否是某一种类型的对象,要注意，后面的数据类型大小写要写对

基本数据类型是不能用这个方法的。

这个方法是在对象的原型链上找，找到就返回true，有继承的时候不是很适用。

//console.log(g instanceof Array) // true

3，construcor 这个方法和instanceof很像，(但是这个方法的返回值是个function（）｛｝)

但是可以处理基本数据类型number string boolean，检测不了object。

这个就是检测构造函数，null和undefind这两个函数被浏览器保护起来了（null和undefind检测会出错）

如果直接写 xxx.construcor 会返回function xxx(){} xxx就是具体的数据类型、

---------------

4,prototype 这个方法几乎是万能的，我们一般用这个方法。

object.prototype.toString();

要区分一下

number.prototype.toString();是转化字符串的

object.prototype.toString();是输出数据类型的，

他的意思是返回当前方法（this指向的）所属类

一般我们用

object.prototype.toString.call();

原理，先调用object对象上的toString方法，然后用cell（）方法改变其中的this指向，

然后就返回了this指向的主体所属的类。

var a = 123,

b = 'test',

c = true,

d = null,

e, // undefind

f = {},

g = [],

h = /\d/, //正则

i = function(){};

console.log(typeof a);// number

console.log(typeof b);// string

console.log(typeof c);// boolean

console.log(typeof d);// object

console.log(typeof e);// undefined

console.log(typeof f);// object

console.log(typeof g);// object

console.log(typeof h);// object

console.log(typeof i);// function

-----------------

console.log(g instanceof Array) // true

console.log(h instanceof RegExp) // true

console.log(i instanceof Function) //true

-----------------

console.log(b.constructor); //function String() { [native code] }

console.log(b.constructor === String);

------------

function testData (option) { //完美准确的识别我们能见到的数据类型

var xx = Object.prototype.toString.call(option);

console.log(xx);

}

testData (a);

testData (b);

testData (c);

testData (d);

testData (e);

testData (f);

testData (g);

testData (h);

testData (i);