## js的数据类型有哪些?

string,number,boolean,undefined,object,null,symbol

## 哪些值会被转化为false?

undefined ,null ,false ,0, NaN ,""(空字符串)

## typeof的返回值有哪些?

1.typeof运算符返回6种类型: undefined,string,number,boolean,object,function

2.typeof返回的结果的类型是字符串类型.即typeof(typeof(123)) == "string"

3.typeof null 的结果是"object"

4.typeof Object的结果也是"object"

5.typeof function a(){}的结果是"function"

## 出现值为undefined的几种情况

1.变量声明了但是没赋值 解析器会给一个默认值 就是undefined.

  var a; console.log(a); // undefined

2. 数组中 某一项没有值,那么值就是undefined.  
    var arr = [1, 2, 3]; console.log(arr[10]); // undefined

3. 形参接收不到值 接收到的就是undefined  
    function fn(a, b) {  
        console.log(b);  
    }  
    fn(1); // undefined

4. 函数没有返回值 相当于返回了undefined,即不写return或者return不写返回值,则函数执行默认返回的是undefined.

5. 对象没有这个属性 非要获取这个属性的值 这个属性的值也是undefined.

## null和undefined的区别

null： null类型，代表“空值”，代表一个空对象指针，使用typeof运算得到 “object”，它是一个特殊的对象值。

undefined： Undefined类型，当一个声明了一个变量未初始化时，得到的就是undefined。

null 和 undefined 都表示“值的空缺”，你可以认为undefined是表示系统级的、出乎意料的或类似错误的值的空缺，而null是表示程序级的、正常的或在意料之中的值的空缺。

null == undefined 为true

null === undefined 为false

null隐式转化为0,undefined转换为NaN

## 判断数据类型有几种方法?

1.typeof

可以判断数据类型，它返回表示数据类型的字符串（返回结果只能包括number,boolean,string,function,object,undefined）；

可以使用typeof判断变量是否存在（如if(typeof a!="undefined"){...}）；

typeof 运算符的问题是无论引用的对象是什么类型它都返回object

2. instanceof

因为A instanceof B 可以判断A是不是B的实例，返回一个布尔值，由构造类型判断出数据类型:

console.log(arr instanceof Array ); // true

console.log(date instanceof Date ); // true

console.log(fn instanceof Function ); // true

//注意： instanceof 后面一定要是对象类型，大小写不能写错，该方法试用一些条件选择或分支.

3.通过Object下的toString.call()方法来判断

Object.prototype.toString.call();

console.log(toString.call(123)); //[object Number]

console.log(toString.call('123')); //[object String]

console.log(toString.call(undefined)); //[object Undefined]

console.log(toString.call(true)); //[object Boolean]

console.log(toString.call(null));//[ object Null]

console.log(toString.call({})); //[object Object]

console.log(toString.call([])); //[object Array]

console.log(toString.call(function(){}))//[object Function]

4. 根据对象的contructor判断

console.log(arr.constructor === Array); //true

console.log(date.constructor === Date); //true

console.log(fn.constructor === Function); //true

5. jq中判断数据类型的方法

jQuery提供了一系列工具方法，用来判断数据类型，以弥补JavaScript原生的typeof运算符的不足:

jQuery.isArray();是否为数组

jQuery.isEmptyObject();是否为空对象 （不含可枚举属性）。

jQuery.isFunction():是否为函数

jQuery.isNumberic():是否为数字

jQuery.isPlainObject():是否为使用“{}”或“new Object”生成对象，而不是浏览器原生提供的对象。

jQuery.isWindow(): 是否为window对象；

jQuery.isXMLDoc(): 判断一个DOM节点是否处于XML文档中。

## typeof方法有什么缺陷?怎么弥补?

typeof数组,对象,null的返回值都是object.解决办法:

使用以上的instanceof,constructor,toString.call()或者:

判断Array 使用Array.isArray(arr)；

判断x是否是null: typeof x== 'object' && x == null

## 请封装一个函数实现isNaN()的功能

思路:isNaN其实是隐式地调用Number()方法转化参数,转成的结果如果是NaN的话返回true,否则返回false

function myIsNaN(x){

if (String(Number(x)) == "NaN"){

return true

}else{

return false

}

}

## 系统的isNaN有什么缺陷?怎么弥补?

我们知道NaN表示一种不合法的数字,它的类型是数字类型,typeof NaN == “number”;而isNaN函数从字面上看就是判断一个数据是否是NaN,它的内部隐式地调用Number对参数进行转化,当转换结果是数字就返回false,否则返回true.

isNaN有一个重大缺陷。它似乎过于按照字面的意思（“不是一个数字”）去理解 NaN 的含义了 —— 它的工作基本上是：“测试这个传进来的东西是否不是一个 number 或者是一个 number”。但这不是十分准确。比如说:

var a = “foo”; isNaN(a); // true

我们知道a是一个字符串,它根本不是NaN,可结果返回的是true, 在 ES6 中，终于提供了一个替代它的工具：Number.isNaN()。可以让你即使是在前 ES6 的浏览器中安全地检查 NaN 值：

Var a = “foo”; Number.isNaN(a); // false

利用js中NaN !== NaN这个唯有NaN具有的特性,我们可以自己很容易实现判断一个数据是否是NaN:

function myIsNaN(n){

return n !== n;

}

## 输出结果

console.log("B" + "a" + + "B" + "a");

答案: "BaNaNa"

## 输出结果

console.log([] == []);

答案: false,因为数组是引用类型,每个[]其实都是两个不同的地址,虽然它们的值是一样的,但是引用类型比较的是地址.

## 输出结果

console.log([] == ![])

答案: true,

![]的结果为false.涉及的知识点是值为false的几种情况.

在判断[] == false的过程中,两边类型不相等,==是宽松等于,所以存在转化类型再比较的情况.false转化为数字为0,[]转化为数字,因为[]是引用类型,转化为number类型要调用valueOf()方法再用Number()方法转数字.所以Number([].valueOf()) = 0.

转化完两边都是0,所以结果为true.

## 输出结果

console.log([] === ![])

答案: false

![]的结果为false.涉及的知识点是值为false的几种情况

严格等于,[] === false,两边类型不一致,结果直接为false.

## 输出结果

var a = "40";var b = 7; a % b 的结果是多少?

答案: 数字类型的 5;字符串的"40" % 7 两边类型不一样,会进行隐式类型转化.

## 输出结果

1.console.log(1+ "2"+"2");

2.console.log(1+ +"2"+"2");

3.console.log("A"- "B"+"2");

4.console.log("A"- "B"+2);

答案:

1. "122",字符串,任何类型的数据加字符串都等于字符串,所以就是字符串拼接,结果为"122"

2. "32",字符串, +"2"会发生隐式类型转化,转化为数字2,所以是1 + 2 + "2",结果为"32"

3. "NaN2",字符串,"A"-"B"为NaN,NaN+"2"结果为"NaN2"

4. NaN,"A" - "B"为NaN,NaN + 2还是NaN

## 输出结果

var x = new Boolean(false);

if (x) {

alert('hi');

}

var y = Boolean(0);

if (y) {

alert('hello');

}

弹出框的结果为: 弹出 hi

var x = new Boolean(false);则x为一个对象,if(x),x转化值为true

var y = Boolean(0);y被转化为false.