1.typeof 返回数据类型

undefined、string、boolean、number、symbol(ES6)、Object、Function

总结：少了null,多了function

//1. 对象和布尔值比较

//对象和布尔值进行比较时，对象先转换为字符串，然后再转换为数字，布尔值直接转换为数字

console.log([] == true); //false []转换为字符串'',然后转换为数字0,true转换为数字1，所以为false

console.log([] == false) false转为0 ，[]转为”” 再转为0

//2. 对象和字符串比较

//对象和字符串进行比较时，对象转换为字符串，然后两者进行比较。

console.log([1,2,3] == '1,2,3'); // true [1,2,3]转化为'1,2,3'，然后和'1,2,3'， so结果为true;

3、如果Infinity与0相乘，结果是NaN

console.dir(5/0);//Infinity

console.dir(0/0);//NaN //0除以0结果是NaN

console.dir(5/null);//Infinity null用Number()函数进行转化，结果是0，那么5/0是正无穷

**其他基本类型转化为字符串**，基本和预期的一样：

[?](https://www.jb51.net/article/103830.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | console.log("" + null);   // "null"  console.log("" + undefined); // "undefined"  console.log("" + false);   // "false"  console.log("" + true);   // "true"  console.log("" + 0);     // "0"  console.log("" + NaN);    // "NaN"  console.log("" + Infinity); // "Infinity" |

**其他基本类型转化为数字**，需要特殊记忆：

[?](https://www.jb51.net/article/103830.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | console.log(+null);     // 0  console.log(+undefined);   // NaN  console.log(+false);     // 0  console.log(+true);     // 1  console.log(+"");      // 0  console.log(+'1');      // 1  console.log(+'1x');     // NaN |

其中null，空字符是0，undefined是NaN。

以上，基本类型转换都说明白了。

**下面来看看引用类型转换为基本类型。**

**引用类型转换为布尔，始终为true**

**引用类型转换为字符串**

1.优先调用toString方法（如果有），看其返回结果是否是原始类型，如果是，转化为字符串，返回。

2.否则，调用valueOf方法（如果有），看其返回结果是否是原始类型，如果是，转化为字符串，返回。

3.其他报错。

**引用类型转化为数字**

1.优先调用valueOf方法（如果有），看其返回结果是否是基本类型，如果是，转化为数字，返回。

2.否则，调用toString方法（如果有），看其返回结果是否是基本类型，如果是，转化为数字，返回。

3.其他报错。

首先我们看看常见引用类型toString和valueOf返回什么？

[?](https://www.jb51.net/article/103830.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | var a = {};  console.dir(a.toString());  // "[object Object]"  console.dir(a.valueOf());  // 对象本身  var b = [1, 2, 3];  console.dir(b.toString());  // "1,2,3"  console.dir(b.valueOf());  // 对象本身  var c = [[1],[2]];  console.dir(c.toString());  // "1,2"  console.dir(c.valueOf());  // 对象本身  var d = function() {return 2};  console.dir(d.toString());  // "function() {return 2}"  console.dir(d.valueOf());  // 对象本身 |

因此对应的转换为字符串和数字的情形是：

[?](https://www.jb51.net/article/103830.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | var a = {};  console.dir(a + "");     // "[object Object]"  console.dir(+a);       // NaN  var b = [1, 2, 3];  console.dir(b + "");     // "1,2,3"  console.dir(+b);       // NaN  var c = [[1],[2]];  console.dir(c + "");     // "1,2"  console.dir(+c);       // NaN  var d = function() {return 2};  console.dir(d + "");     // "function () {return 2}"  console.dir(+d);       // NaN |

再来个报错的情形：

[?](https://www.jb51.net/article/103830.htm)

|  |  |
| --- | --- |
| 1  2  3  4 | var a = {};  a.toString = function() {return {};}  console.log("" + a);     // 报错  console.log(+a)       // 报错 |

以上类型转换规律基本说完。

**最后来说一下万恶的“==”**

面试题如下：

[?](https://www.jb51.net/article/103830.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var a = false;  var b = undefined;  if (a == b) {   alert('true');  } else {   alert('false');  } |

本以为会弹出true的。天那!为啥是false?

哈哈。。。

双等号，如果两边类型不同，会有隐式转换发生。犀牛书75页总结如下：

1，null和undefined，相等。

2，数字和字符串，转化为数字再比较。

3，如果有true或false，转换为1或0，再比较。

4，如果有引用类型，优先调用valueOf。

5，其余都不相等。

因此有：

[?](https://www.jb51.net/article/103830.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5 | console.log(undefined == false); // false  console.log(null == false);   // false  console.log(0 == false);     // true  console.log(NaN == false);    // false  console.log("" == false);    // true |

0 == false之所以为true根据第3条。

"" == false之所以为true根据第3条，变成了"" == 0,再根据第2条。

第4条再来一个例子：

[?](https://www.jb51.net/article/103830.htm)

|  |  |
| --- | --- |
| 1 | console.log([[2]] == '2') |

其上结果为true，原因如下：

[[2]]的valueOf是对象本身，不是基本类型。

尝试调用toString的结果是'2'。

因此变成了'2'和数字2的比较。根据第2条，相等。WTF!!

最后说句，使用"==="就没有这些问题了。

这里又几个巧妙的用法，比如数值可以用 toString() 方法进行[快速的进制转换](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number/toString)；在我前几日写过的一篇小文：[JavaScript绝句小研究](http://www.ituring.com.cn/article/49787)中，绝句6中也展示了一个用 toFixed() 函数进行位数截取的功能。