## 那些诡异的边角知识

### Q1. 运算符考点： 下面程序输出是什么?

console.log(1 < 2 < 3);console.log(3 > 2 > 1);复制代码

输出:

true

flase复制代码

第一个输出结果是好理解的，主要看下第二个为什么是false

核心在于js怎么去解析<和>运算符。 在JS中，这种运算符是从左向右运算的，所以3>2>1就被转换成了true>1,而true的值是1，接着比较1>1就返回false了。

### Q2. typeof，下面输出结果是什么

console.log(typeof typeof 1);复制代码

答案是string

会输出string，这个题目不仅仅是typeof的考察，也是对js运算的一个考察。 在js中一般有两种操作

* 赋值操作，例如a = b 2>3之类的，上面的题目提到过，是从左向右的顺序
* 取值操作， js问内存：有没有见过这个家伙？，比如console.log(a) typeof a 都属于这个类型，是从右向左的

因此，这个题就被分解为typeof 1返回"number"，注意是一个字符串。 接下来typeof "number"，返回string

### Q3. typeof undefined == typeof NULL输出结果是什么

首先搞清楚两点：

* typeof undefined 输出是undefined
* typeof null输出是object

但是，另一方面，因为js对大小写敏感，null ≠ NULL，所以``typeof NULL返回undefined`

结果是： true

### Q4. 递归设计。 实现一个函数，给该函数一个DOM节点，函数访问其所有子元素(所有子元素，不仅仅是直接子元素)，每次访问子元素的时候，并为其传一个callback。

访问一个DOM tree，是一个经典的深度优先搜索的算法

function Traverse(DOM,callback) {

callback(DOM);

var list = DOM.children;

Array.prototype.forEach.apply(list,(item)=>{

Traverse(item,callback); //递归

})

}