链滴

js 中如何准确的判断一个变量的类型?

作者：zhujie
原文链接：https://ld246.com/article/1589958355818
来源网站：链滴

这是一个高频面试题，我们开发中也进程会遇到，今天我们来实现一个函数getValType(val)，用来取一个变量的类型。从1. JS基础变量类型。2. Js中判断变量的函数。 3. 实现getValType函数。3个面来分析实现。

Js基础变量类型

在 JS 中，有 5 种基本数据类型和 1 种复杂数据类型，基本数据类型有：Undefined, Null, Boolean, umber和String；复杂数据类型是Object，Object中还细分了很多具体的类型，比如：Array, Functi n, Date等等。

判断变量的函数

我们先定义一组变量来用下面的函数来测试：

```
var allVarMap = {
    // 数字
    num:123,
    // Infinity
    num1: 1 / 0,
    // NaN
    num2: null / 0,
    // 字符串
    str: 'abcdef',
    // 布尔类型
    bool: true,
    // 数组
    arr :[1, 2, 3, 4],
    // json对象
    json :{name:'wenzi', age:25},
    // 函数
    func:function(){ console.log('this is function'); },
    // 箭头函数
    func1: () => {console.log('arrow function')},
    // undefined类型
    und:undefined,
    // null类型
    nul:null,
    // date类型
    date:new Date(),
    // 正则表达式
    reg :/^[a-zA-Z]{5,20}$/,
    // 异常类型
    error:new Error()
}
```

typeof判断变量类型

typeof运算符用于判断对象的类型，但是对于一些创建的对象，它们都会返回object。我们用该函数判断上面的结果：

```
var results = []
for (let i in allVarMap) {
```

```
    results.push(typeof allVarMap[i])
}
console.log(results.join())
// number,number,number,string,boolean,object,object,function,function,undefined,object,ob
ect,object,object
```

可以看到，NaN和Infinity都检查为number类型，其他的 string,function,boolean,undefined包括头函数都能正确检查出来，但是对于reg,date,null都得到了object。看来还需要具体的判断。

instanceof检查

ECMAScript 引入了另一个 Java 运算符 instanceof 来解决这个问题。instanceof 运算符与 typeof 算符相似，用于识别正在处理的对象的类型。instanceof运算符用来判断一个构造函数的prototype性所指向的对象是否存在另外一个要检测对象的原型链上。与 typeof 方法不同的是，instanceof 方要求开发者明确地确认对象为某特定类型。

```
console.log(allVarMap.date instanceof Date) // true
console.log(allVarMap.func instanceof Function) // true
```

可以看到instanceof可以正确判断出date,func的类型，但是前提是已知该变量的类型，所以这里不合我们的预期。

使用Object.prototype.toString.call

定义：首先，取得对象的一个内部属性[[Class]]，然后依据这个属性，返回一个类似于”[object Array“的字符串作为结果（看过ECMA标准的应该都知道，[[]]用来表示语言内部用到的、外部不可直接访的属性，称为“内部属性”）。利用这个方法，再配合call，我们可以取得任何对象的内部属性[[Class]，然后把类型检测转化为字符串比较，以达到我们的目的。

我们看看下面的函数运行结果：

```
var results = []
for (let i in allVarMap) {
    results.push(Object.prototype.toString.call(allVarMap[i]))
}
console.log(results.join())
// [object Number],[object Number],[object Number],[object String],[object Boolean],[object A
ray],[object Object],[object Function],[object Function],[object Undefined],[object Null],[object
Date],[object RegExp],[object Error]
```

可以看到该函数正确的返回了所有变量的类型，我们只要取出返回结果中的字符串，就能得到变量的型。

实现getValType函数

根据上面的分析，我们可以先用typeOf函数判断出基础类型number,string,function,boolean,undefined。然后如果结果是object，我们再用Object.prototype.toString.call来判断出具体的类型。

```
var getVarType = function (val = 0) {
    var type = typeof val
    // object需要使用Object.prototype.toString.call判断
    if (type === 'object') {
        var typeStr = Object.prototype.toString.call(val)
```

```
        // 解析[object String]
        typeStr = typeStr.split(' ')[1]
        type = typeStr.substring(0, typeStr.length - 1)
    }
    return type
}
var results = []
for (let i in allVarMap) {
    results.push(getVarType(allVarMap[i]))
}
console.log(results.join())
// number,number,number,string,boolean,Array,Object,function,function,number,Null,Date,R
gExp,Error
```

可以看到，完美判断出了所有变量的类型，该函数可以再添加一些逻辑，判断一个变量是否是NaN,Infinity之类的特殊需求。

总结

1. typeOf能判断出一个变量的类型，但是只能判断出number,string,function,boolean,undefined,null和其他对象类型返回结果都为object.
2. instanceof能判断出一个对象是否是另一个类的实例。
3. Object.prototype.toString.call能判断出所有变量的类型，返回值为[Object ***]。