数据类型概述

《JavaScript 教程》作者：阮一峰

1 简介

JavaScript 中的每一个值，都属于某一种数据类型。

JavaScript 的数据类型，共有六种：

- Number
- String
- Boolean
- Undefined
- Null
- Object

ES6 新增了一种类型：

- Symbol

通常，Number，String ，Boolean这三种类型，合称为原始类型（primitive type）的值，即它们是最基本的数据类型，不能再细分了。

至于 Undefined 和 Null，一般将它们看成两个特殊值。

Object 则称为合成类型（complex type）的值，因为一个对象往往是多个原始类型的值的合成，可以看作是一个存放各种值的容器。

对象是最复杂的数据类型，又可以分成多个子类型。

- Object
  - Object
  - Function
  - Array
  - Date
  - RegExp

2 判断一个值的类型

JavaScript 有三种方法，可以确定一个值到底是什么类型：

- typeof 运算符
- instanceof 运算符
- Object.prototype.toString 方法

1. typeof 运算符

   各类型值的 typeof 返回值：

   - Number，返回 "number"
   - String，返回 "string"
   - Boolean，返回 "boolean"
   - Undefined，返回 "undefined"
   - Null，返回 "object"
   - Object
     - Object，返回 "object"
     - Function，返回 "function"
     - Array，返回 "object"
     - Date，返回
     - RegExp，返回

   Null 返回是 object，这是由于历史原因造成的。

   使用 typeof 运算符无法区分 Array 和 Object。

   实例：

```
typeof 123 // "number"
typeof '123' // "string"
typeof false // "boolean"
typeof null // "object"

typeof {} // "object"
typeof [] // "object"

function f() {}
typeof f
// "function"
```

```
typeof undefined
// "undefined"
```

`typeof undefined` 返回 `undefined`。利用这一点，可以用 typeof 来检查一个变量是否声明，而避免报错。

```
if (typeof v === "undefined") {
  // ...
}
```

2. instanceof 运算符

   可以用来区分 Array 和 Object 。

```
var o = {};
var a = [];

o instanceof Array // false
a instanceof Array // true
```

3. Object.prototype.toString 方法

   使用该方法能准确判断出每个值的类型。

   不同数据类型的 Object.prototype.toString 方法返回值如下。

   - Number：返回 `[object Number]`。
   - String：返回 `[object String]`。
   - Boolean：返回 `[object Boolean]`。
   - undefined：返回 `[object Undefined]`。
   - null：返回 `[object Null]`。
   - Object
     - Object: 返回 `[object Object]`。
     - Array：返回 `[object Array]`。
     - Function：返回 `[object Function]`。
     - Date：返回 `[object Date]`。
     - RegExp：返回 `[object RegExp]`。
     - Arguments：返回 `[object Arguments]`。
     - Error：返回 `[object Error]`。

由于实例对象可能会自定义 toString 方法，覆盖掉 Object.prototype.toString 方法，所以为了得到类型字符串，最好直接使用 Object.prototype.toString 方法。

使用函数的 call 方法，可以在任意值上调用这个方法，帮助我们判断这个值的类型。

```
const toString = Object.prototype.toString;

toString.call(2) // "[object Number]"
toString.call('') // "[object String]"
toString.call(true) // "[object Boolean]"
toString.call(undefined) // "[object Undefined]"
toString.call(null) // "[object Null]"
toString.call(Math) // "[object Math]"
toString.call({}) // "[object Object]"
toString.call([]) // "[object Array]"
```