## 数据类型

### JavaScript 中的数据类型

JavaScript 所有的值（数据）都有着自己的**数据类型**。目前 JavaScript 中共有以下的数据类型：

* number（数值）
* bitInt （任意精度整数）
* string（字符串）
* object（对象）
* function（函数）
* undefined
* regexp（正则表达式）
* symbol（符号）

我们已经接触了 number 用于数值计算，string 用于处理文本内容，object 作为存放其他数据的结构，还有 undefined，它自身也是一种独立的数据类型。你可能会为看不见 array 而感到奇怪，事实上，数组是一种特殊的对象，它被精心设计以用于处理数据，但是我们仍需将它归类为 object。至于另外三种数据类型，我们会在后面逐渐接触到。

JavaScript 提供了 typeof 操作符来确定一个值的数据类型，它得到类型的字符串名称，即以上七种之一。

alert(typeof 123); // "number"  
alert(typeof NaN); // "number  
alert(typeof Infinity); // "number"  
alert(typeof "Hello world"); // "string"  
alert(typeof {}); // "object"  
alert(typeof [1, 2, 3]); // "object"  
alert(typeof null); // "object"  
alert(typeof alert); // "function"  
alert(typeof undefined); // "undefined"

Note: 为什么 typeof null === "object" ?

在 JavaScript 第一个版本的实现中，每个值在内部存储时都会用一个标记来记录它的数据类型。由于 object 是 JavaScript 中的“第一等类型”，它的类型标记为 0，而 null 表示为 *NULL 指针*，在大多数平台上， NULL 指针的实际值是 0x00，那么 null 的数据类型标记实际上与 object 相同。因此 typeof 运算符在获取 null 的类型标记的时候，得到的是 0 ，便会将它判断为 "object"。

### 更好的应用

typeof 的首要作用自然是判断数据类型，以便了解到的数据类型做出可能的操作。它具有明显的局限性：它对数据类型的判断仅限于以上七种；无法区分 array 和一般的 object；当我们明确地需要一个**对象**时它却会将 null 混为一谈。我们需要在 typeof 的判断的基础上使用一些辅助方法，以便在各类数据类型之间游刃有余。

1. 区分数组与对象

* typeof 会将数组判定为 "object"，没关系，JavaScript 提供了 Array.isArray 函数来判断一个值是不是数组。那么我们只需要：
* alert(Array.isArray([1, 2, 3])); // true  
  alert(Array.isArray([])); // true  
  alert(Array.isArray(new Array())); // true  
  alert(Array.isArray({})); // false

1. 精确地判断对象

* 我们在第一节（逻辑）中了解过，null 会被当做一个假值，因此对它进行非运算会得到 true。
* 我们可以采取这样的策略：
  + 使用typeof 进行判断，如果它不是 "object"，得到 false。
  + 如果它是 "object"，判断它是否为数组，若是，得到 false。
  + 如果它是 null，得到 false，否则为 true。可以对它进行 !!操作，若为 null ，我们就会得到 false，否则我们就会得到 true。
* 示例：
* let value = "Hello world";  
  alert( // 对 "Hello" 进行判断  
   typeof value !== "object" ? false :  
   Array.isArray(value) ? false :  
   !!value  
  ); // false  
  // value = true -> false  
  // value = 123 -> false  
  // value = null -> false  
  // value = [1, 2, 3] -> false  
  // value = {a: 1, b: 2} -> true  
  // value = window -> true
  1. 精确判断数字
  + NaN 和 ±Infinity 都属于 number 类型，但是我们在进行数学运算的时候并不希望将它们参与到运算中。我们只需要进行一些附加判断即可将它们与真正的数字区分开来。
  + let n = 100;  
    alert(  
     typeof n !== "number" ? false :  
     isNaN(n) ? false : isFinite(n)  
    ); // true  
    // n = "123" -> false  
    // n = NaN -> false  
    // n = Infinity -> false  
    // n = Number.MAX\_VALUE -> true  
    // n = Number.MAX\_VALUE \* 10 -> false