链滴

再谈 JS 中的数据类型以及转换

作者：xiluotop
原文链接：https://ld246.com/article/1571714829600
来源网站：链滴

JS 中的数据类型

- 基本数据类型：
  - Number
  - Boolean
  - String
  - undefined
  - null
  - Symbol (ES6)
  - BigInt (ES6)
- 复杂引用数据类型：
  - Object：诸如内置对象 window,Math,console; Array,Date,RegExp 等
  - Function：诸如内置方法 Array,Date,RegExp 等

使用 typeof 检测数据类型

typeof 多用于检测基本数据类型，如果不需要精准检测的话，也可以用来判断是否是对象或函数，是 null 要除外，因为这是个历史 bug

```
var t1 = 1;
var t2 = '2';
var t3 = false;
var t4 = undefined;
var t5 = null;
```

```
var t6 = Symbol('symbol');
var t7 = BigInt(7);
var t8 = function() {};
var t9 = new Object();
console.log(typeof t1);
console.log(typeof t2);
console.log(typeof t3);
console.log(typeof t4);
console.log(typeof t5);
console.log(typeof t6);
console.log(typeof t7);
console.log(typeof t8);
console.log(typeof t9);
```

```
number
string
boolean
undefined
object
symbol
bigint
function
object
```

null 检测后是 object 注意下就行

再谈 == 与 === 区别

● ===：全等，先判断两边类型是否相同，相同后再判断值是否相同，都相同则返回 true，否则返回 alse，使用时简单，不用担心一些隐式的转换

● ==：此判断不像 === 全等那么严格，在判断时可能会进行隐式的转换，然后再比较值是否相同，隐式转换遵循以下规则：

● 如果两边类型相等，则直接判断其内容是否相同，如 1==2 返回 false,如果是引用复杂类型，比较的是地址 如 obj1 == obj2 返回 false

● 当判断值是否为 null 或 undefined 时，直接进行判断，是的话就返回 true

● 当判断类型为 String 或 Number 时，则会发生隐式转换，会将 String 转换为 Number 后再行判断

● 判断时有一边为 Boolean 类型时，则将其先转换为 Number 类型，然后再进行判断

● 如果判断时其中一个为 Object 类型，而另一边是 String、Number、Symbol 类型，则会将 Object 类型先转换为字符串，然后进行比较

```
console.log('true' == true)  // false，先将 true 转换为 1，然后 'true' 转换为 NaN 与 1 相比，果为 false
console.log({1:1} == true) // false
console.log({1:1} == '[object Object]') // true
// object 与 String，Number，Boolean相比，先转换为字符串(toString)然后进行比较
```

● 总结，由此可见 == 在比较时可能会经常要考虑其隐式转换，所以强烈建议使用全等（===）进行较

对象转换基本数据类型的流程

上面 == 与 === 比较的介绍过程中，最后提到 Object 与基本数据类型比较也有个转化的过程，Object 转换为 基本数据类型也是有个基本流程，其流程如下：

对象转基本数据类型会先调用其内置 toPrimitive 方法，对于其逻辑处理有以下步骤：

- 如果存在 Symbol.toPrimitive 方法，则优先调用并返回数据
- 否则调用 valueOf()，如果可以转换为基本数据类型则返回结果
- 否则调用 toString()，如果可以转换为基本数据类型则返回结果
- 如果以上步骤均无法转换为基本数据类型，则会报错

重写方法转换为基本数据类型：

```
var obj = {

    valueOf() {
        return 1;
    },
    toString() {
        return '2';
    },
    [Symbol.toPrimitive]() {
        return 3;
    }
}
console.log(obj == 1);
 // false
console.log(obj == 2);
 // false
console.log(obj == 3); // true
```

重写方法让其不能转换为基本数据类型：

```
var obj = {
    valueOf() {
        return {};
    },
    toString() {
        return {};
    },
    [Symbol.toPrimitive]() {
        return {};
    }
}
console.log(obj == 1);
```

```
Uncaught TypeError: Cannot convert object to primitive value
    at 数据类型.html:59
```

如何使 if(a==1 && a==2) 条件成立

原文链接：再谈 JS 中的数据类型以及转换

利用上面的对象隐式转换，可以实现这个需求。

当对象与基本数据类型相比，它会调用内置 toPrimitive 进行规则转换，利用这个特性重写相关方法以实现。

```
var a = {
    value : 0,
    [Symbol.toPrimitive](){
        this.value++;
        console.log(this.value);
        return this.value;
    }
}
console.log(a==1 && a==2);
```

```
1
2
true
>
```