null, undefined 和布尔值

null 和 undefined

概述

`null`与`undefined`都可以表示“没有”，含义非常相似。将一个变量赋值为`undefined`或`null`，老实说，语法效果几乎没区别。

```javascript
var a = undefined;
// 或者
var a = null;
```

上面代码中，变量`a`分别被赋值为`undefined`和`null`，这两种写法的效果几乎等价。

在`if`语句中，它们都会被自动转为`false`，相等运算符（`==`）甚至直接报告两者相等。

```javascript
if (!undefined) {
  console.log('undefined is false');
}
// undefined is false

if (!null) {
  console.log('null is false');
}
// null is false

undefined == null
// true
```

从上面代码可见，两者的行为是何等相似！谷歌公司开发的 JavaScript 语言的替代品 Dart 语言，就明确规定只有`null`，没有`undefined`！

既然含义与用法都差不多，为什么要同时设置两个这样的值，这不是无端增加复杂度，令初学者困扰吗？这与历史原因有关。

1995年 JavaScript 诞生时，最初像 Java 一样，只设置了`null`表示"无"。根据 C 语言的传统，`null`可以自动转为`0`。

```javascript
Number(null) // 0
5 + null // 5
```

上面代码中，`null`转为数字时，自动变成0。

但是，JavaScript 的设计者 Brendan Eich，觉得这样做还不够。首先，第一版的 JavaScript 里面，`null`就像在 Java 里一样，被当成一个对象，Brendan Eich 觉得表示“无”的值最好不是对象。其次，那时的 JavaScript 不包括错误处理机制，Brendan Eich 觉得，如果`null`自动转为0，很不容易发现错误。

因此，他又设计了一个`undefined`。区别是这样的：`null`是一个表示“空”的对象，转为数值时为`0`；`undefined`是一个表示"此处无定义"的原始值，转为数值时为`NaN`。

```javascript
Number(undefined) // NaN
5 + undefined // NaN
```

用法和含义

对于`null`和`undefined`，大致可以像下面这样理解。

`null`表示空值，即该处的值现在为空。调用函数时，某个参数未设置任何值，这时就可以传入`null`，表示该参数为空。比如，某个函数接受引擎抛出的错误作为参数，如果运行过程中未出错，那么这个参数就会传入`null`，表示未发生错误。

`undefined`表示“未定义”，下面是返回`undefined`的典型场景。

```javascript
// 变量声明了，但没有赋值
var i;
i // undefined

// 调用函数时，应该提供的参数没有提供，该参数等于 undefined
function f(x) {
  return x;
}
f() // undefined

// 对象没有赋值的属性
var  o = new Object();
o.p // undefined

// 函数没有返回值时，默认返回 undefined
function f() {}
f() // undefined
```

布尔值

布尔值代表“真”和“假”两个状态。“真”用关键字`true`表示，“假”用关键字`false`表示。布尔值只有这两个值。

下列运算符会返回布尔值：

- 前置逻辑运算符： `!` (Not)
- 相等运算符：`===`，`!==`，`==`，`!=`
- 比较运算符：`>`，`>=`，`<`，`<=`

如果 JavaScript 预期某个位置应该是布尔值，会将该位置上现有的值自动转为布尔值。转换规则是除了下面六个值被转为`false`，其他值都视为`true`。

- `undefined`
- `null`
- `false`
- `0`
- `NaN`
- `""`或`''`（空字符串）

布尔值往往用于程序流程的控制，请看一个例子。

```javascript
if ('') {
  console.log('true');
}
// 没有任何输出
```

上面代码中，`if`命令后面的判断条件，预期应该是一个布尔值，所以 JavaScript 自动将空字符串，转为布尔值`false`，导致程序不会进入代码块，所以没有任何输出。

注意，空数组（`[]`）和空对象（`{}`）对应的布尔值，都是`true`。

```javascript
if ([]) {
  console.log('true');
}
// true

if ({}) {
  console.log('true');
}
// true
```

更多关于数据类型转换的介绍，参见《数据类型转换》一章。

参考链接

- Axel Rauschmayer, [Categorizing values in JavaScript](http://www.2ality.com/2013/01/categorizing-values.html)