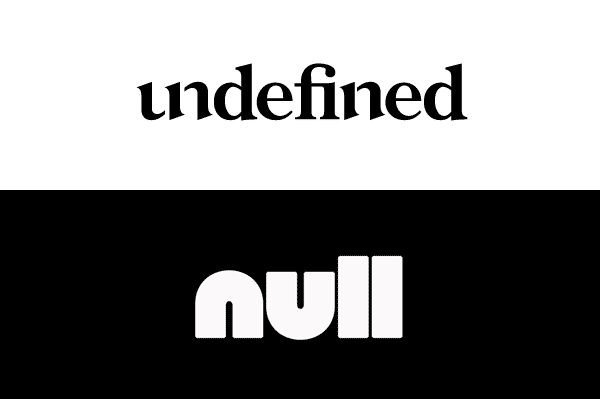
# undefined与null的区别

大多数计算机语言，有且仅有一个表示"无"的值，比如，C语言的NULL，Java语言的null，Python语言的None，Ruby语言的nil。

有点奇怪的是，JavaScript语言居然有**两个**表示"无"的值：undefined和null。这是为什么？



## 一、相似性

在JavaScript中，将一个变量赋值为undefined或null，老实说，几乎没区别。

var a = undefined;

var a = null;

上面代码中，a变量分别被赋值为undefined和null，这两种写法几乎等价。

undefined和null在if语句中，都会被自动转为false，相等运算符甚至直接报告两者相等。

if (!undefined)

console.log('undefined is false');

// undefined is false

if (!null)

console.log('null is false');

// null is false

undefined == null

// true

上面代码说明，两者的行为是何等相似！

既然undefined和null的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JavaScript的复杂度，令初学者困扰吗？Google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！

## 二、历史原因

最近，我在读新书[《Speaking JavaScript》](http://speakingjs.com/)时，意外发现了这个问题的答案！

原来，这与JavaScript的历史有关。1995年[JavaScript诞生](http://www.ruanyifeng.com/blog/2011/06/birth_of_javascript.html)时，最初像Java一样，只设置了null作为表示"无"的值。

根据C语言的传统，null被设计成可以自动转为0。

Number(null)

// 0

5 + null

// 5

但是，JavaScript的设计者Brendan Eich，觉得这样做还不够，有两个原因。

首先，null像在Java里一样，被当成一个对象。但是，JavaScript的数据类型分成原始类型（primitive）和合成类型（complex）两大类，Brendan Eich觉得表示"无"的值最好不是对象。

其次，JavaScript的最初版本没有包括错误处理机制，发生数据类型不匹配时，往往是自动转换类型或者默默地失败。Brendan Eich觉得，如果null自动转为0，很不容易发现错误。

因此，Brendan Eich又设计了一个undefined。

## 三、最初设计

JavaScript的最初版本是这样区分的：**null是一个表示"无"的对象，转为数值时为0；undefined是一个表示"无"的原始值，转为数值时为NaN。**

Number(undefined)

// NaN

5 + undefined

// NaN

## 四、目前的用法

但是，上面这样的区分，在实践中很快就被证明不可行。目前，null和undefined基本是同义的，只有一些细微的差别。

**null表示"没有对象"，即该处不应该有值。**典型用法是：

（1） 作为函数的参数，表示该函数的参数不是对象。

（2） 作为对象原型链的终点。

Object.getPrototypeOf(Object.prototype)

// null

**undefined表示"缺少值"，就是此处应该有一个值，但是还没有定义。**典型用法是：

（1）变量被声明了，但没有赋值时，就等于undefined。

（2) 调用函数时，应该提供的参数没有提供，该参数等于undefined。

（3）对象没有赋值的属性，该属性的值为undefined。

（4）函数没有返回值时，默认返回undefined。

var i;

i // undefined

function f(x){console.log(x)}

f() // undefined

var o = new Object();

o.p // undefined

var x = f();

x // undefined