Object.prototype.valueOf()

`valueOf()` 方法返回指定对象的原始值。

语法

```
object.valueOf()
```

返回值

返回值为该对象的原始值。

描述

JavaScript调用 `valueOf` 方法将对象转换为原始值。你很少需要自己调用 `valueOf` 方法；当遇到要预期的原始值的对象时，JavaScript会自动调用它。

默认情况下， `valueOf` 方法由 `Object` 后面的每个对象继承。 每个内置的核心对象都会覆盖此方法以返回适当的值。如果对象没有原始值，则 `valueOf` 将返回对象本身。

你可以在自己的代码中使用 `valueOf` 将内置对象转换为原始值。 创建自定义对象时，可以覆盖 `Object.prototype.valueOf()` 来调用自定义方法，而不是默认 `Object` 方法。

覆盖自定义对象的 valueOf 方法

你可以创建一个取代 `valueOf` 方法的函数，你的方法必须不能传入参数。

假设你有个对象叫 `MyNumberType` 而你想为它创建一个 `valueOf` 方法。下面的代码为 `valueOf` 方法赋予了一个自定义函数：

```
MyNumberType.prototype.valueOf = function() { return customPrimitiveValue
```

有了这样的一个方法，下一次每当 `MyNumberType` 要被转换为原始类型值时，JavaScript 在此之前会自动调用自定义的 `valueOf` 方法。

`valueOf` 方法一般都会被 JavaScript 自动调用，但你也可以像下面代码那样自己调用：

```
1 myNumberType.valueOf()
```

注意：字符串上下文中的对象通过 `toString()` 方法转换，这与使用 `valueOf` 转换为原始字符串的 `String` 对象不同。所有对象都能转换成一个“`[object 类型]`”这种格式的字符串。但是很多对象不能转换为数字，布尔或函数。

示例

使用 valueOf

```
1  function myNumberType(n) {
2      this.number = n;
3  }
4
5  myNumberType.prototype.valueOf = function() {
6      return this.number;
7  };
8
9  myObj = new myNumberType(4);
10 myObj + 3; // 7
```

规范

Specification	Status	Comment
ECMAScript 1st Edition (ECMA-262)	ST Standard	Initial definition. Implemented in JavaScript 1.1.
ECMAScript 5.1 (ECMA-262) Object.prototype.valueOf	ST Standard	
ECMAScript 2015 (6th Edition, ECMA-262) Object.prototype.valueOf	ST Standard	
ECMAScript Latest Draft (ECMA-262) Object.prototype.valueOf	LS Living Standard	

浏览器兼容

Desktop Mobile

Feature	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari
Basic Support	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)

参考

- `Object.prototype.toString()`
- `parseInt()`