Object

Object 构造函数创建一个对象包装器。

语法

```
// 对象初始化器（Object initialiser）或对象字面量（literal）
{[nameValuePair1[, nameValuePair2[, ...nameValuePairN]]]}

// 以构造函数形式来调用
new Object([value])
```

参数

nameValuePair1, nameValuePair2, ... nameValuePairN
成对的名称（字符串）与值（任何值），其中名称通过冒号与值分隔。

value
任何值。

描述

Object 构造函数为给定值创建一个对象包装器。如果给定值是 null 或 undefined，将会创建并返回一个空对象，否则，将返回一个与给定值对应类型的对象。

当以非构造函数形式被调用时， Object 等同于 new Object() 。

可查看 对象初始化/字面量语法。

Object 构造函数的属性

Object.length
值为1。

Object.prototype
可以为所有 Object 类型的对象添加属性。

Object 构造函数的方法

Object.assign()
通过复制一个或多个对象来创建一个新的对象。

Object.create()
使用指定的原型对象和属性创建一个新对象。

Object.defineProperty()
给对象添加一个属性并指定该属性的配置。

Object.defineProperties()
给对象添加多个属性并分别指定它们的配置。

Object.entries()
返回给定对象自身可枚举属性的 `[key, value]` 数组。

Object.freeze()
冻结对象：其他代码不能删除或更改任何属性。

Object.getOwnPropertyDescriptor()
返回对象指定的属性配置。

Object.getOwnPropertyNames()
返回一个数组，它包含了指定对象所有的可枚举或不可枚举的属性名。

Object.getOwnPropertySymbols()
返回一个数组，它包含了指定对象自身所有的符号属性。

Object.getPrototypeOf()
返回指定对象的原型对象。

Object.is()
比较两个值是否相同。所有 NaN 值都相等（这与==和===不同）。

Object.isExtensible()
判断对象是否可扩展。

Object.isFrozen()
判断对象是否已经冻结。

Object.isSealed()

判断对象是否已经密封。

`Object.keys()`

返回一个包含所有给定对象自身可枚举属性名称的数组。

`Object.preventExtensions()`

防止对象的任何扩展。

`Object.seal()`

防止其他代码删除对象的属性。

`Object.setPrototypeOf()`

设置对象的原型（即内部 `[[Prototype]]` 属性）。

`Object.values()`

返回给定对象自身可枚举值的数组。

`Object` 实例和 `Object` 原型对象

JavaScript中的所有对象都来自 `Object`；所有对象从 `Object.prototype` 继承方法和属性，尽管它们可能被覆盖。例如，其他构造函数的原型将覆盖 `constructor` 属性并提供自己的 `toString()` 方法。 `Object` 原型对象的更改将传播到所有对象，除非受到这些更改的属性和方法将沿原型链进一步覆盖。

属性

`Object.prototype.constructor`

特定的函数，用于创建一个对象的原型。

`Object.prototype.__proto__` ⚠

指向当对象被实例化的时候，用作原型的对象。

`Object.prototype.__noSuchMethod__` ⚠

当未定义的对象成员被调用作方法的时候，允许定义并执行的函数。

`Object.prototype.__count__`

用于直接返回用户定义的对象中可数的属性的数量。已被废除。

`Object.prototype.__parent__`

用于指向对象的内容。已被废除。

方法

Object.prototype.__defineGetter__() ⚠👎

关联一个函数到一个属性。访问该函数时，执行该函数并返回其返回值。

Object.prototype.__defineSetter__() ⚠👎

关联一个函数到一个属性。设置该函数时，执行该修改属性的函数。

Object.prototype.__lookupGetter__() ⚠👎

返回使用 `__defineGetter__` 定义的方法函数 。

Object.prototype.__lookupSetter__() ⚠👎

返回使用 `__defineSetter__` 定义的方法函数。

Object.prototype.hasOwnProperty()

返回一个布尔值 ，表示某个对象是否含有指定的属性，而且此属性非原型链继承的。

Object.prototype.isPrototypeOf()

返回一个布尔值，表示指定的对象是否在本对象的原型链中。

Object.prototype.propertyIsEnumerable()

判断指定属性是否可枚举，内部属性设置参见 ECMAScript DontEnum attribute 。

Object.prototype.toSource() ⚠

返回字符串表示此对象的源代码形式，可以使用此字符串生成一个新的相同的对象。

Object.prototype.toLocaleString()

直接调用 `toString()` 方法。

Object.prototype.toString()

返回对象的字符串表示。

Object.prototype.unwatch() ⚠

移除对象某个属性的监听。

Object.prototype.valueOf()

返回指定对象的原始值。

Object.prototype.watch() ⚠

给对象的某个属性增加监听。

Object.prototype.eval() 🗑

在指定对象为上下文情况下执行javascript字符串代码，已经废弃。

示例

给定 undefined 和 null 类型使用 Object

下面的例子将一个空的 Object 对象存到 o 中：

```
var o = new Object();
```

```
var o = new Object(undefined);
```

```
var o = new Object(null);
```

使用 Object 生成布尔对象

下面的例子将 Boolean 对象存到 o 中：

```
// 等价于 o = new Boolean(true);
var o = new Object(true);
```

```
// 等价于 o = new Boolean(false);
var o = new Object(Boolean());
```

规范

Specification	Status	Comment
ECMAScript 1st Edition (ECMA-262)	ST Standard	Initial definition. Implemented in JavaScript 1.0.
ECMAScript 5.1 (ECMA-262) Object	ST Standard	
ECMAScript 2015 (6th Edition, ECMA-262) Object	ST Standard	Added Object.assign, Object.getOwnPropertySymbols, Object.setPrototypeOf, Object.is
ECMAScript Latest Draft (ECMA-262)	LS Living	Added Object.entries and Object.values.

Object	Standard	

浏览器兼容

Desktop Mobile

Feature	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari
Basic Support	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)
`prototype`	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)
`__count__`	No	No	No	No	No	No
`__noSuchMethod__`	No	No	(Yes) — 43	No	No	No
`__parent__`	No	No	No	No	No	No
`__proto__`	(Yes)	(Yes)	(Yes)	11	(Yes)	(Yes)
`constructor`	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)
`assign`	45	(Yes)	34	No	32	9
`create`	5	(Yes)	4	9	11.6	5
`defineProperties`	5	(Yes)	4	9	11.6	5
`defineProperty`	5	(Yes)	4	9[1]	11.6	5.1[2]
`entries`	54	(Yes)	47	No	No	10.1
`freeze`	6	(Yes)	4	9	12	5.1
`getNotifier`	36 — 52	No	No	No	No	No
`getOwnPropertyDescriptor`	5	(Yes)	4	8	12	5
`getOwnPropertyDescriptors`	54	(Yes)	50	No	41	10
`getOwnPropertyNames`	5	(Yes)	4	9	12	5
`getOwnPropertySymbols`	38	(Yes)	36	No	25	9
`getPrototypeOf`	5	(Yes)	3.5	9	12.10	5
`is`	30	(Yes)	22	No	(Yes)	9
`isExtensible`	6	(Yes)	4	9	12	5.1
`isFrozen`	6	(Yes)	4	9	12	5.1
`isSealed`	6	(Yes)	4	9	12	5.1
`keys`	5	(Yes)	4	9	12	5
`observe`	36 — 52	No	No	No	No	No
`preventExtensions`	6	(Yes)	4	9	12	5.1
`__defineGetter__`	(Yes)	(Yes)	(Yes)[3]	11	(Yes)	(Yes)
`__defineSetter__`	(Yes)	(Yes)	(Yes)[3]	11	(Yes)	(Yes)
`__lookupGetter__`	(Yes)	(Yes)	(Yes)	11	(Yes)	(Yes)
`__lookupSetter__`	(Yes)	(Yes)	(Yes)	11	(Yes)	(Yes)

`eval`	No	No	No	No	No	No
`hasOwnProperty`	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)
`isPrototypeOf`	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)
`propertyIsEnumerable`	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)
`toLocaleString`	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)
`toSource`	No	No	(Yes)	No	No	No
`toString`	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)
`unwatch`	No	No	(Yes)	No	No	No
`valueOf`	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)
`watch`	No	No	(Yes)	No	No	No
`seal`	6	(Yes)	4	9	12	5.1
`setPrototypeOf`	34	(Yes)	31	11	(Yes)	9
`unobserve`	36 — 52	No	No	No	No	No
`values`	54	(Yes)	47	No	(Yes)	10.1

1. Also supported in Internet Explorer 8, but only on DOM objects and with some non-standard behaviors.

2. Also supported in Safari 5, but not on DOM objects.

3. Starting with Firefox 48, this method can no longer be called at the global scope without any object. A `TypeError` will be thrown otherwise. Previously, the global object was used in these cases automatically, but this is no longer the case.

相关连接

- 初始化对象