Object.prototype.__proto__

> **警告:** 通过现代浏览器的操作属性的便利性，可以改变一个对象的 `[[Prototype]]` 属性, 这种行为在每一个JavaScript引擎和浏览器中都是一个非常慢且影响性能的操作，使用这种方式来改变和继承属性是对性能影响非常严重的，并且性能消耗的时间也不是简单的花费在 `obj.__proto__ = ...` 语句上, 它还会影响到所有继承来自该 `[[Prototype]]` 的对象，如果你关心性能，你就不应该在一个对象中修改它的[[Prototype]].。相反, 创建一个新的且可以继承 `[[Prototype]]` 的对象，推荐使用 `Object.create()` 。

> **警告:** 当 `Object.prototype.__proto__` 已被大多数浏览器厂商所支持的今天，其存在和确切行为仅在ECMAScript 2015规范中被标准化为传统功能，以确保Web浏览器的兼容性。为了更好的支持，建议只使用 `Object.getPrototypeOf()` 。

`Object.prototype` 的 `__proto__` 属性是一个访问器属性（一个getter函数和一个setter函数），暴露了通过它访问的对象的内部 `[[Prototype]]` (一个对象或 `null`)。

使用__proto__是有争议的，而且是不鼓励的。 它从来没有被包括在EcmaScript语言规范中，但是现代浏览器实现了它, 无论如何。__proto__属性已在ECMAScript 6语言规范中标准化，用于确保Web浏览器的兼容性，因此它未来将被支持。它已被弃用, 赞成 `Object.getPrototypeOf` / `Reflect.getPrototypeOf` 和 `Object.setPrototypeOf` / `Reflect.setPrototypeOf` （尽管如此，设置对象的[[Prototype]]是一个缓慢的操作，如果性能是一个问题，应该避免）。

__proto__ 属性也可以在对象文字定义中使用对象[[Prototype]]来创建，作为 `Object.create()` 的一个替代。 请参阅： object initializer / literal syntax.

语法

```
let shape = {},
    circle = new Circle();

// 设置该对象的原型链引用
// 过时且不推荐使用的。这里只是举个栗子， 尽量不要在生产环境中这样做。
shape.__proto__ = circle;

```

```
// 判断该对象的原型链引用是否属于circle
console.log(shape.__proto__ === circle); // true
```

```
let shape = function () {};
let p = {
    a: function () {
        console.log('aaa');
    }
};
shape.prototype.__proto__ = p;

let circle = new shape();

circle.a();//aaa

console.log(shape.prototype === circle.__proto__);//true

//或者

let shape = function () {
};
var p = {
    a: function () {
        console.log('a');
    }
};

let circle = new shape();
circle.__proto__ = p;


circle.a(); //  a

console.log(shape.prototype === circle.__proto__);//false

//或者

function test() {
}
test.prototype.myname = function () {
    console.log('myname');

```

```
}
var a = new test()

console.log(a.__proto__ === test.prototype);//true

a.myname();//myname


//或者

var fn = function () {
};
fn.prototype.myname = function () {
    console.log('myname');
}

var obj = {
    __proto__: fn.prototype
};


obj.myname();//myname
```

注意：这是两个下划线，后面是五个字符的 “proto”，后面再跟两个下划线。

描述

__proto__的读取器(getter)暴露了一个对象的内部 `[[Prototype]]` 。对于使用对象字面量创建的对象，这个值是 `Object.prototype` 。对于使用数组字面量创建的对象，这个值是 `Array.prototype` 。对于functions，这个值是 `Function.prototype` 。对于使用 new fun 创建的对象，其中fun是由js提供的内建构造器函数之一(`Array` , `Boolean` , `Date` , `Number` , `Object` , `String` 等等），这个值总是fun.prototype。对于用js定义的其他js构造器函数创建的对象，这个值就是该构造器函数的prototype属性。

__proto__ 的设置器(setter)允许对象的 `[[Prototype]]`被变更。前提是这个对象必须通过 `Object.isExtensible()` : 进行扩展，如果不这样，一个 `TypeError` 错误将被抛出。要变更的值必须是一个object或 `null` ，提供其它值将不起任何作用。

要理解原型如何被使用，请查看相关文章：Inheritance and the prototype chain。

.__proto__属性是 `Object.prototype` 一个简单的访问器属性，其中包含了get（获取）和set（设置）的方法，任何一个__proto__的存取属性都继承于 `Object.prototype` ，但一个访问属性如果不

是来源于 `Object.prototype` 就不拥有.__proto__属性，譬如一个元素设置了其他的.__proto__属性在 `Object.prototype` 之前，将会覆盖原有的 `Object.prototype` 。

规范

Specification	Status	Comment
ECMAScript 2015 (6th Edition, ECMA-262) Object.prototype.__proto__	ST Standard	Included in the (normative) annex for additional ECMAScript legacy features for Web browsers (note that the specification codifies what is already in implementations).
ECMAScript Latest Draft (ECMA-262) Object.prototype.__proto__	LS Living Standard	

浏览器兼容情况

Desktop Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
Basic support	(Yes)	(Yes)	11	(Yes)	(Yes)

兼容性注意事项

在 ECMAScript 2015（ES6）的规范要求中，支持__proto__ 是各大Web浏览器厂商的要求（虽然符合规范），但其他环境下因为历史遗留的问题，也有可能被使用和支持。

更多请参考

- `Object.prototype.isPrototypeOf()`
- `Object.getPrototypeOf()`
- `Object.setPrototypeOf()`