**instanceof**

**instanceof** **运算符**用于检测构造函数的 prototype 属性是否出现在某个实例对象的原型链上。

[**语法**](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/instanceof#syntax)

*object* instanceof *constructor*

[参数](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/instanceof#parameters)

**object**

某个实例对象

**constructor**

某个构造函数

[**描述**](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/instanceof#description)

instanceof 运算符用来检测 constructor.prototype 是否存在于参数 object 的原型链上。

// 定义构造函数

function C(){}

function D(){}

var o = new C();

o instanceof C; // true，因为 Object.getPrototypeOf(o) === C.prototype

o instanceof D; // false，因为 D.prototype 不在 o 的原型链上

o instanceof Object; // true，因为 Object.prototype.isPrototypeOf(o) 返回 true

C.prototype instanceof Object // true，同上

C.prototype = {};

var o2 = new C();

o2 instanceof C; // true

o instanceof C; // false，C.prototype 指向了一个空对象,这个空对象不在 o 的原型链上.

D.prototype = new C(); // 继承

var o3 = new D();

o3 instanceof D; // true

o3 instanceof C; // true 因为 C.prototype 现在在 o3 的原型链上

Copy to Clipboard

需要注意的是，如果表达式 obj instanceof Foo 返回 true，则并不意味着该表达式会永远返回 true，因为 Foo.prototype 属性的值有可能会改变，改变之后的值很有可能不存在于 obj 的原型链上，这时原表达式的值就会成为 false。另外一种情况下，原表达式的值也会改变，就是改变对象 obj 的原型链的情况，虽然在目前的ES规范中，我们只能读取对象的原型而不能改变它，但借助于非标准的 \_\_proto\_\_ 伪属性，是可以实现的。比如执行 obj.\_\_proto\_\_ = {} 之后，obj instanceof Foo 就会返回 false 了。

[instanceof 和多全局对象(例如：多个 frame 或多个 window 之间的交互)](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/instanceof#instanceof_%E5%92%8C%E5%A4%9A%E5%85%A8%E5%B1%80%E5%AF%B9%E8%B1%A1%E4%BE%8B%E5%A6%82%EF%BC%9A%E5%A4%9A%E4%B8%AA_frame_%E6%88%96%E5%A4%9A%E4%B8%AA_window_%E4%B9%8B%E9%97%B4%E7%9A%84%E4%BA%A4%E4%BA%92)

在浏览器中，我们的脚本可能需要在多个窗口之间进行交互。多个窗口意味着多个全局环境，不同的全局环境拥有不同的全局对象，从而拥有不同的内置类型构造函数。这可能会引发一些问题。比如，表达式 [] instanceof window.frames[0].Array 会返回 false，因为 Array.prototype !== window.frames[0].Array.prototype，并且数组从前者继承。

起初，你会认为这样并没有意义，但是当你在你的脚本中开始处理多个 frame 或多个 window 以及通过函数将对象从一个窗口传到另一个窗口时，这就是一个有效而强大的话题。比如，实际上你可以通过使用Array.isArray(myObj) 或者Object.prototype.toString.call(myObj) === "[object Array]" 来安全的检测传过来的对象是否是一个数组。

比如检测一个 Nodes 在另一个窗口中是不是 SVGElement，你可以使用myNode instanceof myNode.ownerDocument.defaultView.SVGElement

**Mozilla开发者注意:**  
在代码中使用 XPCOM instanceof 有特殊影响: 如果查询接口成功执行后，obj instanceof *xpcomInterface* (e.g. Components.interfaces.nsIFile) 调用obj.QueryInterface(*xpcomInterface*) 并且返回 true 。这种调用的副作用是在一次成功的 instanceof 测试后，你可以在 obj 上使用xpcomInterface 的属性。这与标准的 JavaScript 全局变量不同，即使 obj 来自不同的作用域，obj instanceof xpcomInterface 也可以按预期产生作用。

[**示例**](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/instanceof#examples)

[演示 String 对象和 Date 对象都属于 Object 类型和一些特殊情况](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/instanceof#example_demonstrating_that_string_and_date_are_of_type_object)

下面的代码使用了 instanceof 来证明：String 和 Date 对象同时也属于Object 类型（他们是由 Object 类派生出来的）。

但是，使用对象文字符号创建的对象在这里是一个例外：虽然原型未定义，但 instanceof Object 返回 true。

var simpleStr = "This is a simple string";

var myString = new String();

var newStr = new String("String created with constructor");

var myDate = new Date();

var myObj = {};

var myNonObj = Object.create(null);

simpleStr instanceof String; // 返回 false, 非对象实例，因此返回 false

myString instanceof String; // 返回 true

newStr instanceof String; // 返回 true

myString instanceof Object; // 返回 true

myObj instanceof Object; // 返回 true, 尽管原型没有定义

({}) instanceof Object; // 返回 true, 同上

myNonObj instanceof Object; // 返回 false, 一种创建非 Object 实例的对象的方法

myString instanceof Date; //返回 false

myDate instanceof Date; // 返回 true

myDate instanceof Object; // 返回 true

myDate instanceof String; // 返回 false

Copy to Clipboard

[演示 mycar 属于 Car 类型的同时又属于 Object 类型](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/instanceof#example_demonstrating_that_mycar_is_of_type_car_and_type_object)

下面的代码创建了一个类型 Car，以及该类型的对象实例 mycar. instanceof 运算符表明了这个 mycar 对象既属于 Car 类型，又属于 Object 类型。

function Car(make, model, year) {

this.make = make;

this.model = model;

this.year = year;

}

var mycar = new Car("Honda", "Accord", 1998);

var a = mycar instanceof Car; // 返回 true

var b = mycar instanceof Object; // 返回 true

[不是...的实例](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/instanceof#%E4%B8%8D%E6%98%AF...%E7%9A%84%E5%AE%9E%E4%BE%8B)

要检测对象不是某个构造函数的实例时，你可以这样做

if (!(mycar instanceof Car)) {

// Do something, like mycar = new Car(mycar)

}

这和以下代码完全不同

if (!mycar instanceof Car)

Copy to Clipboard

这段代码永远会得到 false（!mycar 将在 instanceof 之前被处理，所以你总是在验证一个布尔值是否是 Car 的一个实例）。