1. **Object.assign()** 方法用于将所有可枚举属性的值从一个或多个源对象复制到目标对象。它将返回目标对象。

const object1 = {

a: 1,

b: 2,

c: 3

};

const object2 = Object.assign({c: 4, d: 5}, object1);

console.log(object2.c, object2.d);

// expected output: 3 5

1. Object.create()方法创建一个新对象，使用现有的对象来提供新创建的对象的\_\_proto\_\_。 （请查看浏览器控制台以获取视觉证据。）

const person = {

isHuman: false,

printIntroduction: function () {

console.log(`My name is ${this.name}. Am I human? ${this.isHuman}`);

}

};

const me = Object.create(person);

me.name = "Matthew"; // "name" is a property set on "me", but not on "person"

me.isHuman = true; // inherited properties can be overwritten

me.printIntroduction();

// expected output: "My name is Matthew. Am I human? true"

1. **Object.getPrototypeOf()** 方法返回指定对象的原型（内部[[Prototype]]属性的值）。

### 参数

**obj**

要返回其原型的对象。

### 返回值

给定对象的原型。如果没有继承属性，则返回 [null](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/null" \o "值 null 特指对象的值未设置。它是 JavaScript 基本类型 之一。) 。

var proto = {};var obj = Object.create(proto);

Object.getPrototypeOf(obj) === proto; // true

var reg = /a/;

Object.getPrototypeOf(reg) === RegExp.prototype; // true

1. **Object.setPrototypeOf()**方法设置一个指定的对象的原型 ( 即, 内部[[Prototype]]属性）到另一个对象或  [null](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/null" \o "值 null 特指对象的值未设置。它是 JavaScript 基本类型 之一。)。

## 语法

Object.setPrototypeOf(obj, prototype)

### 参数

**obj**

要设置其原型的对象。.

**prototype**

该对象的新原型(一个对象 或 [null](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/null" \o "值 null 特指对象的值未设置。它是 JavaScript 基本类型 之一。)).

var dict = Object.setPrototypeOf({}, null);

1. **isPrototypeOf()** 方法用于测试一个对象是否存在于另一个对象的原型链上。

isPrototypeOf() 与 [instanceof](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/instanceof" \o "instanceof 运算符用来测试一个对象在其原型链中是否存在一个构造函数的 prototype 属性。) 运算符不同。在表达式 "object instanceof AFunction"中，object 的原型链是针对 AFunction.prototype 进行检查的，而不是针对 AFunction 本身。

## 语法

*prototypeObj*.isPrototypeOf(*object*)

### 参数

**object**

在该对象的原型链上搜寻

### 返回值

[Boolean](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Boolean" \o "此页面仍未被本地化, 期待您的翻译!)，表示调用对象是否在另一个对象的原型链上。

### 报错

**[TypeError](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/TypeError" \o "TypeError（类型错误） 对象用来表示值的类型非预期类型时发生的错误。)**

如果 prototypeObj 为 undefined 或 null，会抛出 [TypeError](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/TypeError" \o "TypeError（类型错误） 对象用来表示值的类型非预期类型时发生的错误。)。

function Foo() {}function Bar() {}function Baz() {}

Bar.prototype = Object.create(Foo.prototype);

Baz.prototype = Object.create(Bar.prototype);

var baz = new Baz();

console.log(Baz.prototype.isPrototypeOf(baz)); // true

console.log(Bar.prototype.isPrototypeOf(baz)); // true

console.log(Foo.prototype.isPrototypeOf(baz)); // true

console.log(Object.prototype.isPrototypeOf(baz)); // true