Object.defineProperty() 方法会直接在一个对象上定义一个新属性，或者修改一个对象的现有属性， 并返回这个对象。

语法：

Object.defineProperty(obj, prop, descriptor)

参数

obj

要在其上定义属性的对象。

prop

要定义或修改的属性的名称。

descriptor

将被定义或修改的属性描述符。

返回值

被传递给函数的对象。

**描述 descriptor**

该方法允许精确添加或修改对象的属性。通过赋值来添加的普通属性会创建在属性枚举期间显示的属性（for...in 或 Object.keys 方法）， 这些值可以被改变，也可以被删除。这种方法允许这些额外的细节从默认值改变。默认情况下，使用Object.defineProperty()添加的属性值是不可变的。

对象里目前存在的属性描述符有两种主要形式：数据描述符和存取描述符。数据描述符是一个具有值的属性，该值可能是可写的，也可能不是可写的。访问器描述符是由getter-setter函数对描述的属性。描述符必须是这两种形式之一；不能同时是两者。

数据描述符和存取描述符均具有以下可选键值：

configurable

当且仅当该属性的 configurable 为 true 时，该属性描述符才能够被改变，同时该属性也能从对应的对象上被删除。默认为 false。False表明一旦对象属性被创建和设置相关的属性描述后就不能再次被修改或重复创建，但是writeable属性除外。

enumerable

当且仅当该属性的enumerable为true时，该属性才能够出现在对象的枚举属性中。默认为 false。如果使用直接赋值的方式创建对象的属性，则这个属性的enumerable为true，如o.d = 4 ，此时d可被枚举。

数据描述符同时具有以下可选键值：

value

该属性对应的值。可以是任何有效的 JavaScript 值（数值，对象，函数等）。默认为 undefined。

writable

当且仅当该属性的writable为true时，value才能被赋值运算符改变。默认为 false。

存取描述符同时具有以下可选键值：

get

一个给属性提供 getter 的方法，如果没有 getter 则为 undefined。该方法返回值被用作属性值。默认为 undefined。

set

一个给属性提供 setter 的方法，如果没有 setter 则为 undefined。该方法将接受唯一参数，并将该参数的新值分配给该属性。默认为 undefined。

如果一个描述符同时设置了value,writable,get和set关键字，那么它将被认为是一个数据描述符。如果一个描述符同时有value或writable和get或set关键字，将会产生一个异常，因为set和get是设置或取值，此时在配置value或导致默认的set或get是取配置的function返回值还是配置的value值呢？见示例2。

记住，这些选项不一定是自身属性，如果是继承来的也要考虑。为了确认保留这些默认值，你可能要在这之前冻结 Object.prototype，明确指定所有的选项，或者将\_\_proto\_\_属性指向null。

**创建属性**

如果对象中不存在指定的属性，Object.defineProperty()就创建这个属性。当描述符中省略某些字段时，这些字段将使用它们的默认值。拥有布尔值的字段的默认值都是false。value，get和set字段的默认值为undefined。一个没有get/set/value/writable定义的属性被称为“通用的”，并被“键入”为一个数据描述符。

代码示例：

function Archiver() {

var temperature = null;

var archive = [];

Object.defineProperty(this, 'temperature', {

get: function() {

console.log('get!');

return temperature;

},

set: function(value) {

temperature = value;

archive.push({ val: temperature });

}

});

this.getArchive = function() { return archive; };

}

var arc = new Archiver();

arc.temperature; // 'get!'

arc.temperature = 11;

arc.temperature = 13;

arc.getArchive(); // [{ val: 11 }, { val: 13 }]

示例2，set、get返回非value属性值

var pattern = {

get: function () {

return 'I alway return this string,whatever you have assigned';

},

set: function () {

this.myname = 'this is my name string';

}

};

function TestDefineSetAndGet() {

Object.defineProperty(this, 'myproperty', pattern);

}

var instance = new TestDefineSetAndGet();

instance.myproperty = 'test';

// 'I alway return this string,whatever you have assigned'

console.log(instance.myproperty);

// 'this is my name string'

console.log(instance.myname);