const 扩展（扩展方法）总结

一、 数据属性描述符汇总如下：

数据属性描述器构成

	属性	值类型	默认值	含义
数据性质	value	任意类型	undefined	基本的名/值
	writable	Boolean	false	属性可否重写，默认是只读
	enumerable	Boolean	false	属性可否枚举，默认不能被 for in 枚举（循环遍历）
	configurable	Boolean	false	属性可否被重新配置，默认 writable 和 enumerable 可修改,且用 delete 删除

二、Object.defineProperty( )函数和 Object.hasOwnProperty( )函数

1、Object.defineProperty( )函数

（1）概念：将属性添加到对象上，或修改现有属性的特性

（2）语法：

Object.defineProperty(object, propertyname, descriptor)

（3）参数：

Object ：必需。被添加或修改属性的对象。

propertyname：必需。一个包含属性名称的字符串。

descriptor：必需。属性描述符。它可以针对数据属性或访问器属性。

（4）返回值：

已修改的对象。

（5）用法：

向对象添加新属性。当对象不具有指定的属性名称时，发生此操作。

修改现有属性的特征。当对象已具有指定的属性名称时，发生此操作。

（6）举例：

- 设置一个新属性。

  代码如下：

```
var obj = {name : 'John'};
// 设置一个新的属性
Object.defineProperty(obj, "newDataProperty",{
    value:101
});
console.log(obj);
```

  结果：可以设置新属性。

```
▶ {name: "John", newDataProperty: 101}
```

- 设置新属性并修改新属性的值

  代码如下：

```
var obj = { name: 'John' };

// 设置一个新属性并设置可修改
Object.defineProperty(obj, "newDataProperty", {
    value: 101,
    writable: true
});

//修改新属性的值
obj.newDataProperty = 102;

console.log("Property value: " + obj.newDataProperty);
```

结果：设置了 writable 属性为 true 之后才可以修改新属性的值。

```
Property value: 102
```

2、Object.hasOwnProperty( )函数

（1）概念： 检测是否是原型链中的属性，如果是则返回 false，不是则返回 true（也就是自己自定义的属性）。

（2）语法：object.hasOwnProperty(propertyName)

（3）参数： propertyName：String 类型，指定的属性名称。

（4）例子：

- 是否可以检测出自定义的属性和方法

代码如下：

```
// 定义一个对象并设置属性
const Person = {
    name: '小明',
    say: function() {
        console.log('说话啦');
    }
}

// 打印是否存在name属性
console.log("name:" + Person.hasOwnProperty("name"));
// 打印是否存在say方法
console.log("say:" + Person.hasOwnProperty("say"));
```

结果：可以检测出自定义存在的属性和方法

```
name:true
say:true
```

- 是否可以检测出原型链中定义的属性和方法

代码如下：

```
// 定义一个对象并设置属性
const Person = {
    name: '小明',
    say: function() {
        console.log('说话啦');
    }
}

// 在原型链中添加一个属性

Person.__proto__.age = 20;
// 在原型链中添加一个方法
Person.prototype = {
    run: function() {
        console.log('跑步啦');
    }
}
// 打印是否存在name属性
console.log("name:" + Person.hasOwnProperty("name"));
// 打印是否存在say方法
console.log("say:" + Person.hasOwnProperty("say"));
// 打印是否存在age属性
console.log("age:" + Person.hasOwnProperty("age"));
// 打印是否存在run方法
console.log("run:" + Person.hasOwnProperty("run"));
```

结果：只能检测出自定义属性和方法，不能检测出原型链中的属性和方法。

```
name:true
say:true
age:false
run:false
```

三、对象的密封，冻结

1、Object.seal( ) 密封对象

（1）概念：让一个对象密封，并返回被密封后的对象。密封对象是指那些不能添加新的属性，不能删除已有的属性，以及不能修改已有属性的可枚举性、可配置性、可写性，但是可以修改已有属性对象的值。

（2）例子：

- 添加一个新的属性，修改已有属性的值，删除已有的属性。

  代码如下：

```
const obj = {
    name: 'John',
    sex: 'Lady'
}

//添加一个新属性
obj.age = 30;
// 修改已有属性的值
obj.name = "Rose";

// 删除已有属性
delete obj.sex;    delete删除一个元素，这是es6的写法。
// 打印obj对象
console.log(obj);
```

  结果 :可以添加一个新的属性 ,可以修改已有的属性 ,可以删除已有的属性。

```
▶ {name: "Rose", age: 30}
```

- 密封对象并添加一个新的属性

  代码如下：

```
const obj = { name: 'John' }
//密封
Object.seal(obj);

// 添加一个新属性
obj.age = 30;
console.log("obj.age:" + obj.age);
```

  结果：密封对象之后就不能添加一个新的属性了。

```
obj.age:undefined
```

- 密封对象并修改已有属性的值

代码如下：

```
var obj = { name: 'John' }

// 密封
Object.seal(obj)

// 可以修改已有属性的值
obj.name = 'Backus';
console.log(obj.name);
```

结果：密封对象之后可以修改已有属性的值。

```
Backus
```

- 密封对象并删除已有属性的值。

代码如下：

```
const obj = { name: 'John' }

// 密封
Object.seal(obj);

// 删除已有属性
delete obj.name
// 打印obj对象
console.log(obj)
```

结果：密封对象之后不能删除已有的属性。

```
▶ {name: "John"}
```

- 密封对象并修改已有属性的可枚举性、可配置性、可写性。

代码如下：

```
var obj = { name: 'John' }

// 密封
Object.seal(obj)

// 修改已有的配置属性
Object.defineProperty(obj, 'name', {
    configurable: true,
    writable: true,
    enumerable: true
})
```

结果：浏览器提示报错，密封对象之后不能修改已有属性的可枚举性、可配置性、可写性。

```
▶Uncaught TypeError: Cannot redefine property: name
    at Function.defineProperty (<anonymous>)
    at 2.html:17
```

2、Object.freeze( )冻结对象

（1）概念：这个方法比 Object.seal( )更厉害 ,冻结对象是指那些不能添加新属性，不能修改已有属性的值，不能删除已有属性，以及不能修改已有属性的可枚举性、可配置性、可写性的对象。也就是说，这个对象永远是不可改变的。

（2）例子：只测试与 Object.seal( )不同的地方，也就是不能修改已有属性的值

- 冻结对象并修改已有属性的值

代码如下:

```
var obj = { name: 'John' }
//冻结
Object.freeze(obj)
//修改已有属性的值
obj.name = 'Backus'
console.log('obj.name:' + obj.name)
```

结果：冻结对象之后不能修改已有属性的值。

```
obj.name:John
```