对象

对象引用

如果不同的变量名指向同一个对象，那么它们都是这个对象的引用，也就是说指向同一个内存地址。修改其中一个变量，会影响到其他所有变量。

```
let o1 = {};
let o2 = o1;

o1.a = 1;
o2.a // 1

o2.b = 2;
o1.b // 2
```

如果取消某一个变量对于原对象的引用，不会影响到另一个变量。

```
let o1 = {};
let o2 = o1;

o1 = 1;
o2 // {}
```

这种引用只局限于对象，如果两个变量指向同一个原始类型的值。那么，变量这时都是值的拷贝。

```
let x = 1;
let y = x;

x = 2;
y // 1
```

属性读取

数字键可以不加引号，因为会自动转成字符串。

```
let obj = {
  0.7: 'Hello World'
};

obj['0.7'] // "Hello World"
obj[0.7] // "Hello World"
```

数值键名不能使用点运算符（因为会被当成小数点），只能使用方括号运算符。

```
let obj = {
  123: 'hello world'
};

obj.123 // 报错
obj[123] // "hello world"
```

属性删除

删除一个不存在的属性，`delete` 不报错，而且返回 `true`。

```
let obj = {};
delete obj.p // true
```

属性是否存在

（1） `in` 运算符

```
let obj = { p: 1 };
'p' in obj // true
'toString' in obj // true
```

对象 `obj` 本身并没有 `toString` 属性，但是 `in` 运算符会返回 `true`，因为这个属性是继承的。

（2） `hasOwnProperty` 方法

`hasOwnProperty` 方法判断是否是对象自身的属性。

```
let obj3 = {a: 1, b: 2};
'a' in obj3  // true
'toString' in obj3  // true
obj3.hasOwnProperty('toString')  // false

let obj4 = {'c': 4, 'toString': 5}
obj4.hasOwnProperty('toString');  //  true
```