对象

Object类型，我们也称为一个对象。是JavaScript中的引用数据类型。

它是一种复合值，它将很多值聚合到一起，可以通过名字访问这些值。对象也可以看做是属性的无序集合，每个属性都是一个名/值对。

对象除了可以创建自有属性，还可以通过从一个名为原型的对象那里
继承属性。

除了字符串、数字、true、false、null和undefined之外（es6中新增Symbol，es2021新增Bigint），JS中的值
都是对象。

创建对象

```
// 第一种，使用new命令新建一个对象
var person = new Object();
person.name = 'demacia';
person.age = 19;

// 第二种，直接赋值
var person = {
    name: 'demacia',
    age: 18
}
```

键名

对象的所有键名都是字符串（ES6 又引入了 Symbol 值也可以作为键名），所以加不加引号都可以

对象的存储

JavaScript在运行时数据是保存到`栈内存`和`堆内存`当中的。

简单来说栈内存用来保存变量和基本类型。堆内存用来保存对象。

我们在声明一个变量时实际上就是在栈内存中创建了一个空间用来保存变量。

如果是基本类型则在栈内存中直接保存，如果是引用类型则会在堆内存中保存，变量中保存的实际上对象在堆内存中的地址。

对象的引用

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。

```
var obj1 = {};
var obj2 = obj1;
obj2.name = 'foo';
console.log(obj1.name); // => foo
obj2 = {};
console.log(obj1.name); // => foo
```

基本数据类型和引用数据类型

JS中一共有7种基本数据类型：String、Number、Boolean、Undefined、Null、Symbol、BigInt。

基本数据类型的值是无法修改的，是不可变的。

当一个变量是一个对象时，实际上变量中保存的并不是对象本身，而是对象的引用。

```
var a = 123;
var b = true;
var c = "hello";
var d = {name:'demacia',age:18};
```

变量名	栈内存	堆内存地址	堆内存
d	0x000	0x000	name = 'demacia',age = 18
c	hello		
b	true		
a	123		

表达式还是语句?

对象采用大括号表示，这导致了一个问题：如果行首是一个大括号，它到底是表达式还是语句?

```
{ foo: 123 }
```

- 这是一个表达式，表示一个包含foo属性的对象；
- 这是一个语句，表示一个代码区块，里面有一个标签foo，指向表达式123。

为了避免这种歧义，JavaScript 引擎的做法是，如果遇到这种情况，无法确定是对象还是代码块，一律解释为代码块。

如果要解释为对象，最好在大括号前加上圆括号。因为圆括号的里面，只能是表达式，所以确保大括号只能解释为对象。

属性的操作

读取

读取对象的属性，有两种方法，一种是使用点运算符，还有一种是使用方括号运算符。

```
var obj = {
  p: 'Hello World'
};

obj.p // "Hello World"
obj['p'] // "Hello World"
```

请注意，如果使用方括号运算符，键名必须放在引号里面，否则会被当作变量处理。

```
var foo = 'bar';

var obj = {
  foo: 1,
  bar: 2
};

obj.foo  // 1
obj[foo]  // 2
```

赋值

点运算符和方括号运算符，不仅可以用来读取值，还可以用来赋值。

```
var obj = {};

obj.foo = 'Hello';
obj['bar'] = 'World';
```

JavaScript 允许属性的“后绑定”，也就是说，你可以在任意时刻新增属性，没必要在定义对象的时候，就定义好属性。

删除

`delete`命令用于删除对象的属性，删除成功后返回`true`。

```
var obj = { p: 1 };
Object.keys(obj); // ["p"]

delete obj.p; // true
obj.p; // undefined
console.log(obj);
```

删除一个不存在的属性会返回true。

只有一种情况，delete命令会返回false，那就是该属性存在，且不得删除。

存在

in运算符用于检查对象是否包含某个属性（注意，检查的是键名，不是键值），如果包含就返回true，否则返回false。它的左边是一个字符串，表示属性名，右边是一个对象。

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

in运算符的一个问题是，它不能识别哪些属性是对象自身的，哪些属性是继承的。

使用`hasOwnProperty`可以判断是否是自身属性。

属性查看

for...in

for...in循环用来遍历一个对象的全部属性。

```
var obj = {a: 1, b: 2, c: 3};

for (var i in obj) {
  console.log('键名：', i);
  console.log('键值：', obj[i]);
}
```

for...in循环有两个使用注意点。

- 它遍历的是对象所有可遍历（enumerable）的属性，会跳过不可遍历的属性。
- 它不仅遍历对象自身的属性，还遍历继承的属性。

Object.keys

查看一个对象本身的所有属性，可以使用Object.keys方法。

```
var obj = {
  key1: 1,
  key2: 2
};

Object.keys(obj);
// ['key1', 'key2']
```

with 语句

```
with (对象) {
  语句;
}
```

它的作用是操作同一个对象的多个属性时，提供一些书写的方便。

```
// 例一
var obj = {
  p1: 1,
  p2: 2,
};
with (obj) {
  p1 = 4;
  p2 = 5;
}
// 等同于
obj.p1 = 4;
obj.p2 = 5;

// 例二
with (document.links[0]){
  console.log(href);
  console.log(title);
```

```
  console.log(style);
}
// 等同于
console.log(document.links[0].href);
console.log(document.links[0].title);
console.log(document.links[0].style);
```

注意，如果*with*区块内部有变量的赋值操作，必须是当前对象已经存在的属性，否则会创造一个当前作用域的全局变量。