《深入理解JavaScript》第17章相关内容

JS中的面向对象（OOP）分为几层：第一层：单一对象的面对对象；第二层：对象间的原型链；第三层：作为实例工厂的构造函数，类似于其他语言中的类；

第四层：子类，通过继承已有的构造函数，创建新的构造函数。

JS中的所有对象都是从字符串到值的映射。

属性的分类：数据属性、访问器属性、内置属性。

数据属性：对象中普通属性，包括方法被称为数据属性。

访问器属性：类似于读、写属性的特殊方法。访问器属性可以计算属性的值，可以看成是虚拟的属性。

内置属性：只存在于ECMAScript语言规范中。它们不能被JS直接访问，但可以间接访问。内置属性的值在[]中。例如：[[Prototype]]持有一个对象的原型，并可以通过Object.getPrototypeOf()访问。

JS的对象字面量可以直接创建简单对象（Object的直接实例）。

ECMAScript5允许对象字面量的最后一个属性之后跟一个逗号，但不是所有浏览器都支持。

获取对象的属性：Object.属性名/Object[]

如果一个属性不存在，给它赋值时会自动创建该属性，如果一个属性已经存在，赋值时会改变该属性的值。

delete可以从一个对象中完全移除一个属性。 delete Object.属性名。如果属性是自有属性，且不能被删除，delete会返回一个false。

保留字可以作为对象属性的名字（键）。例如：var function；数字也可以作为属性的键，会被解析为字符串。

点操作符只能访问键为标识符的属性，中括号操作符来访问键为数字的属性。对象字面量也可以使用任意字符串作为属性的键，但是必须加上引号，用中括号操作符来访问这些属性的值。

中括号操作符会将括号中的内容强制转换为字符串。

把任意值转换为对象：Object可以提供这个服务:无参数调用、undefined、null均会转换为{}。

call(thisValue,arg1,arg2,……)：第一个参数会赋值给被调用函数内的this,剩余的参数传入被调函数。

apply(thisValue,argArray):第一个参数会赋值给被调用函数内的this,第二个参数是一个数组，为被调用函数提供参数。

bind(thisValue,arg1,arg2,……)：创建一个新的函数，并用接下来的方式调用bind()，this的值是thisValue,剩余的参数是新函数的参数。

换句话说：当调用原来的函数时，新函数将其参数追加到arg1,arg2,……，argN的后面。

JS中的（...）三点运算符：可以把数组转换为实际参数。这个操作符可以对数组使用Math.max()。

例如：Math.max(...[13,7,30])等价于：Math.max(13,7,30)

两个对象间的原型关系类似于继承：每个对象都可以把另一个对象作为它的原型，并继承原型的所有属性。

通过[[Prototype]]属性连接成的对象链称为原型链。

在一个原型链中，一个对象的属性可以覆写“之后”对象的相同键的属性：前者的属性被最先找到，它隐藏了后者的属性。这种工作原理类似于基于类的语言中的方法重写。

共享可以解决内存问题。常见的模型：数据保存在原型链的第一对象中，而方法保存在后面的对象中。例：var jane={[[Prototype]]:PersonProto,name:'Jane'};

使用给定prototype创建新对象：调用方法：Object.create(proto,propDescObj?)。创建原型为proto的对象。

读取对象原型：object.getPrototypeOf(obj):返回obj的原型。检查一个对象是否是另一个对象的原型：Object.prototype.isPrototypeof(obj)

\_\_proto\_\_不属于CEMAScript5标准。然而，很多引擎开始支持\_\_proto\_\_,它将成为ECMScritp6的一部分。

设置和删除仅影响自有属性。原型属性可以由多个对象共享，这种方式不会改变原有属性，只是影响当前对象。