对象是javascript的基本数据类型。对象是一种复合值：他将很多值（原始值或者其他对象）聚合在一起，可通过名字访问这些值。对象也可看作是属性的无序集合，每个属性都是一个名/值对。属性名是字符串，因此我们可以把对象看成是字符串到值得映射。然而对象不仅仅是字符串到值得映射，除了可以保持自有属性，javascript对象还可以从一个称为原型的对象继承属性。对象的方法通常是继承的属性。

Javascript对象是动态的—可以新增属性也可以删除属性—但它们常用来模拟静态对象以及静态类型语言中的“结构体”（struct）。有时它们也用作字符串的集合（忽略名/值中的值）。

除了字符串、数字、true、false、null和undefined之外，javascript中的值都是对象。

对象是可变的，我们通过引用来而非值来操作对象。

对象常见的用法是创建（create）、设置（set）、查找（query）、删除（delete）、检测（test）、枚举（enumerate）它的属性。

属性包括名字和值。属性名可以是包含空字符串在内的任意字符串，但对象中不能穿在两个同名的属性。值可以是任意的javascript值，或者（ECMAScript 5中）可以是一个setter或者getter函数（或两者都有）。

除了名字和值之外，每个属性还有一些与之相关的值，称为“属性特性（property attribute）”：

* 可写（writable attribute），表明是否可以设置该属性的值。
* 可枚举（enumerate attribute），表明是否可以通过for/in循环返回该属性。
* 可配置（configurable attribute），表明是否可以删除或修改该属性。

在ECMAScript 5之前，通过代码给对象创建的所有属性都是可写、可枚举的、可配置的。

除了包含属性之外、每个对象还拥有三个相关的对象特性（object attribute）：

* 对象的原型（prototype）指向另外一个对象，本对象的属性继承自它的原型对象。
* 对象的类（class）是一个表示对象类型的字符串。
* 对象的扩展标记（extensible flag）指明了（在ECMAScript 5中）是否可以向该对象添加新属性。

三类javascript对象和两类属性的区分：

* 内置对象（native onject）是由ECMAScript规范定义的对象或类。例如，数组、函数、日期、和正则表达式都是内置对象。
* 宿主对象（host object）是由javascript解释器所嵌入的宿主环境（比如web浏览器）定义的。客户端javascript中表示网页结构的HTMLElement对象均是宿主对象。既然宿主环境定义的方法可以当成普通的javascript对象，那么宿主对象也可以当成内置对象。
* 自定义对象（user-defined object）是由运行中的javascript代码创建的对象。
* 自有属性（own property）是直接在对象中定义的属性
* 继承属性（inherited property）是在对象的原型对象中定义的属性。

在ECMAScript5（以及ECMAScript 3的一些实现）中，保留字可以用做不带引号的属性名。然而对于ECMAScript 3来说，使用保留字作为属性名必须使用引号引起来，在ECMASript中对象直接量中最后一个属性后的逗号将忽略，且在ECMAScript 3的大部分实现中也可以忽略，但是在IE中则报错。

所有通过对象直接量创建的对象都具有同一个原型对象，并且可以通过javascript代码Object.prototype获得对象原型的引用。通过关键字new和构造函数调用创建的对象的原型就是构造函数的prototype属性的值。

没有原型的对象为数不多，Object.prototype就是其中之一。它不继承任何属性。其他原型对象都是普通对象，普通对象都具有原型。所有的内置构造函数（以及大部分自定义的构造函数）都具有一个继承自Object.prototype的原型。例如new Date（）创建的Date对象的属性同时继承自Date.prototype和Object.prototype。这一系列链接的原型对象就是所谓的“原型链”（prototype chain）

属性的查询和设置有两种方法

* object.property 使用点运算符和一个标识符
* object[“property”] 使用方括号和一个字符串

javascript对象都是关联数组

字符串值是动态的，可以在运行时更改标识符是静态的，必须写死的程序中。

Javascript对象具有“自有属性”（own property），也有一些属性是从原型对象继承而来的。

假设要查询对象o的属性x，如果o中不存在x，那么将会继续在o的原型中查找，如果仍然没有找到会继续向对象的原型查找，直到找到x或者一个原型是null的对象为止。

给对象x赋值，如果x中已经有属性x（这个属性不是继承来的），那么赋值操作只改变这个已有属性x的值。如果o中不存在x，那么赋值操作给o添加一个新属性x。如果之前o继承自属性x，那么继承的属性就被新创建的同名属性覆盖。

属性赋值操作首先检查原型链，以判断是否允许赋值操作。如果允许赋值操作，它也总是在原对象上创建属性或对已有的属性赋值，而不会去修改原型链。

在javascript中只有查询属性时才会体会到继承的存在，而设置属性则和继承无关，这个特性可以让程序员有选择的覆盖（override）继承的属性。

属性的访问并不总是返回或设置一个值。

查询一个不存在的属性并不会报错。

如果对象不存在，试图查询不存在对象的属性就会报错。

给null和undefined设置属性会报类型错误。给其他值设置属性也不总是成功，有一些属性是只读的，不能重新赋值，有一些对象不允许新增属性但是这些设置属性的失败操作都不会报错；

在以下情况下给对象o设置属性p会失败

* o中的属性p是只读的：不能给只读属性重新赋值（defineProperty（）方法中有一个例外，可以对可配置的只读属性重新赋值）
* o中的属性p是继承属性，且它是只读的：不能通过同名自有属性覆盖只读的继承属性
* o中不存在自有属性p：o没有使用setter方法继承属性p，并且o的可扩展性（extensible attribute）的false。如果o中不存在p，而且没有setter方法可供调用，则p一定会添加至o中。但如果o是不可扩展的，那么在o中不能定义新属性。

delete 运算符可以删除对象的属性 。

delete只是断开属性和宿主对象的联系，而不会去操作属性中的属性。

delete运算符只能删除自有属性 ，不能删除继承属性（要删除继承属性必须从定义这个属性的原型对象上删除它，而且这会影响到所有继承自这个原型的对象）

当delete表达式删除成功或没有任何副作用（比如删除一个不存在的属性）时，它返回true。

如果delete后不是一个属性访问表达式，delete同样返回true。

delete不能删除那些可配置性为false的属性（尽管可以删除不可扩展对象化的可配置属性）。某些内置对象的属性是不可配置的，比如通过变量声明和函数声明创建的全局对象的属性。在严格模式下，删除一个不可配置属性会报一个类型错误。在非严格模式下（以及ECMAScript 3中），这些情况会返回一个false。

In运算符左侧是属性名（字符串），右侧是对象。如果对象的自有属性或继承属性中包含这个属性则返回ture。

“property” in object； 使用模型

hasOwnproperty()方法用来检测给定的名字是否是对象的自由属性。对于继承属性它将返回false。

object.hasOwnProperty(property) 使用模型

properIsEnumerable()方法只有检测到时自有属性且这个自有属性的可枚举性（enumerable attribute）为true时它才返回true。某些内置属性是不可枚举的。通常javascript代码创建的属性都是可枚举的，除非在ECMAScript 5中使用方法来改变属性的可枚举性。

object.propertyIsEnumerable(property) 使用模型

除了使用in运算符之外，另一种更简便的方法是使用“！==”来判断一个属性是否是undefined；但是这种方法不能区分不存在的属性和存在但值为undefined的属性。”！==“可以区分undefined和null。

for/in循环可以在循环体中遍历对象中所有可枚举的属性(包括自有属性和继承属性)，把属性名称赋值给循环变量。

extend()，把p中可枚举的属性复制到o中并返回o，如果o和p中有同名属性，则覆盖o中的属性

merge(),将p中的可枚举的属性复制到o中，如果o和p中有同名属性，o中的属性不受影响，返回o（这个函数不处理getter和setter以及复制属性）

restrict(),如果o中的属性在p中没有同名属性，则删除o中的这个属性，返回o

subtract(), 如果o中的属性在p中存在同名属性，则从o中删除这个属性，返回o

union(),返回一个新对象，这个对象同时拥有o的属性和p的属性，如果o和p中有同名属性，使用p中的属性，返回o

intersection(),返回一个新对象这个对象拥有同时在o和p中出现的属性，使用o中属性的值。

Keys(),返回一个数组这个数组包含的是o中可枚举的自有属性的名字.

以上7个函数都是自定义的工具函数

在ECMAScript 5中定义了两个用以枚举属性名称的函数。

第一个是Object.keys()，它返回一个数组，这个数组由对象中可枚举的自有属性的名称组成。

第二个是Object.getOwnPropertyNames(),返回对象的所有自有属性，包括不可枚举的属性。

对象属性是由名字、值和一组特性（attribute）构成的。在ECMAScript 5中，属性值可以用一个或是两个方法替代，这两个方法就是getter和setter。由getter和setter定义的属性叫做“存取器属性”（accessor property），它不同于“数据属性”（data property），数据属性只有一个简单的值。

getter方法返回的值就是属性春去表达式的值

setter方法将赋值表达式右侧的值当参数传入setter。

和数据属性一样，存取器属性是可以继承的。

属性除了包含名字和值之外，属性还包括一些标识它们可写、可枚举和可配置的特性。在ECMAScript 3中无法设置这些特性，所有通过ECMAScript 3的程序创建的属性都是可写的、可枚举的、可配置的，且无法对这些特性做修改。

* 可以通过这些API给原型对象添加方法并将他们设置成不可枚举的，让它们看起来更像内置方法
* 可以通过这些API给对象定义不能修改或删除的属性，借此“锁定”这个对象。

数据属性的4个特性分别是它的值（value）、可写性（writable）、可枚举性（enumerable）、可配置性（configurable）。存储器属性不具有值（value）特性和可写性，它们的可写性是由setter方法存在与否决定的。

存储器属性的4个特性是读取（get）、写入（set）、可枚举性和可配置性。

为了实现属性的特性的查询和设置操作，ECMAScript 5中定义了一个“属性描述符”（property descriptor）的对象，这个对象代表那4个特性。

数据属性的描述符对象的属性有value、writable、enumerable和configurable。

存储器属性的描述符对象则是get、set、enumerable和configurable。

其中writable、enumerable、configurable都是布尔值。

通过调用Object.getOwnPropertyDescriptor()可与ihuoqu某个对象特定属性的属性描述符。

调用方法 Object.getOwnPropertyDescriptor(object,property)

Object.getOwnPropertyDescriptor()只能得到自有属性的描述符。要想获取继承属性的特性，需要遍历原型链。

要想设置属性的特性，或者想让新建属性具有某种特性，则需要调用Object.defineProperty()，传入要修改的对象、要创建或修改的属性的名称以及属性描述符对象：

调用方法 Object.defineProperty(object，property，property descriptor)

传入Object.defineProperty()的属性描述符对象不必包含4个所有特性。对于新创建的属性来说，默认的特性值是false或者undefined。对于修改的已有属性来说，默认的特性没有做任何修改。这个方法要么修改已有属性要么新建新建自有属性，但不能修改继承属性。

如果需要同时修改或创建多个属性，则需要使用Obejct.defineProperties()。第一个参数是要修改的对象，第二个参数是一个映射表，它包含要新建或修改的属性名称，以及他们的属性符。

调用方法

Object. defineProperties(object,{ property1:{ property descriptor } ,property2: { property descriptor }})

对于那些不允许创建或修改的属性来说，如果用Object.defineProperty()和Obejct.defineProperties()对其操作（新建或修改）就会抛出类型错误异常。

造成这些方法抛出类型错误异常的其他原因则和特性本身相关。可写性控制着对值特性的修改。可配置性控制着对其他特性（包括属性是否可以删除）的修改。

Object.defineProperty()和Obejct.defineProperties()完整规则

* 如果对象是不可扩展的，则可以编辑已有的自有属性，但不能给它添加新属性
* 如果属性是不可配置的，则不能修改它的可配置性和可枚举性
* 如果存取器属性是不可配置的则不能修改其getter和setter方法，也不能将它转换为数据属性。
* 如果数据属性是不可配置的，则不能将它转为存取器属性。
* 如果数据属性是不可配置的们则不能将它的可写性从false修改为true，但可以从true修改为false
* 如果数据属性是不可配置且不可写，则不能修改它的值。然而可配置但不可写属性的值是可以修改的（实际上是先将它标记为可写，然后修改它的值最后转换为不可写）

对象直接量语法给新对象定义存储器属性，但不能查询属性的getter和setter方法或给已有的对象添加新的存取器属性。在ECMAScript 5中，可以用

Object.getOwnPropertyDescriptor()和Object.defineProperty()来实现

每个对象都有与之相关的原型(prototype)、类(class)、可扩展性（extensible attribute）。

对象的原型属性是用来继承属性的，原型属性是在实例对象创建之初就设置好的

在EMCAScript 5中将对象作为参数传入Object.getPrototypeOf()可以查询它的原型，在EMCAScript 3中，则没有与之等价的函数，但经常使用表达式o.constructor.prototype来检测一个对象的原型。（此方法用来检测对象原型并不可靠）

想要检测一个对象是否是另一个对象的原型（或处于原型链中），使用isPrototypeOf()方法.如果是则返回true反之false

p. isPrototypeOf(o)，来检测p是否是o的原型。

isPrototypeOf()实现的功能 和 instanceof运算符非常类似

对象的类属性（class attribute）是一个字符串，用以表示对象的类型信息。EMCAScript 5和EMCAScript 3都没有设置这个属性的方法，并只有一种方法可以查询它。

通过自定义的函数classof()查询对象的类属性

classof可以传入任何类型的参数。

对象的可扩展性用以表示是否可以给对象添加新属性。所有内置对象和自定义对象都是显式可扩展的，宿主对象的可扩展性是由javascript引擎定义的。

ECMAScript 5定义了用来查询和设置对象可扩展性的函数。通过将对象传入Object.isExtensible()，来判断该对象是否是可扩展的。如果想要将对象转换为不可扩展的，需要调用Object.preventExtensions()，将待转换的对象作为参数传进去。注意一旦转换为不可扩展的，就无法将其转换回可扩展的。Object.preventExtensions()只影响到对象本身的可扩展性。如果给一个不可扩展对象的原型添加新属性，那么这个不可扩展的对象同样会继承这些新属性。

可扩展性的母的是将对象“”锁定，以避免外界的干扰。对象的可扩展性通常和属性的可配置性与可写性配合使用

Object.seal()和Object.preventExtensions()类似，除了能够将对象设置为不可扩展的，还可以将对象的所有自有属性都设置为不可配置的。即不能给这个对象添加新属性。对于那些已经封闭（sealed）起来的对象是不能解封的。可以使用Object.isSealed()来检测对象是否封闭。

Object.freeze()将更严格地锁定对象——“冻结”（frozen）除了将对象设置为不可扩展和将其属性设置为不可配置之外，还可以将它自有的所有数据属性设置为只读（如果对象的存取器属性具有setter方法，存取器属性将不受影响，仍可以通过给属性赋值调用它们）。使用Object.isFrozen()来检测对象是否冻结。

Object.preventExtensions()，Object.seal()，Object.freeze()都返回传入的对象

对象序列化（serialization）是指将对象的状态转换成字符串，也可将字符串还原为对象。

ECMAScript 5提供内置函数JSON.stringify()和JSON.parse()用来序列化和还原javascript对象

在ECMAScript 3中可以通过引用json2.js模块在ECMAScript 3中的环境使用这些函数

NaN、Infinity和-Infinity序列化的结果是null，日期对象序列化的结果是ISO格式的日期字符串（参照Date.toJSON()函数），但JSON.parse()已然保留他们的字符串形态，不会将它们还原成原始日期对象。函数，RegExp、Error对象和undefined值不能序列化和还原。

JSON.stringify()只能序列化对象可枚举的自有属性。对于一个不能序列化的属性来说，在序列化后输出字符串中会将这个属性略掉。JSON.stringify()和JSON.parse()都可以接收第二个可选参数，通过传入需要序列化或还原的属性列表来定制自定义的序列化或还原操作。

Object.toString()方法没有参数，它将返回一个表示调用这个方法的对象值的字符串。

Object.toLocaleString()方法，这个方法返回一个表示这个对象的本地化字符串。

Object中toLocaleString()方法，仅调用toString()方法并返回对应值。Date和Number类对toLocaleString()方法做了定制，可以用它对数字、日期和时间做本地化的转换。Array的toLocaleString()方法很像，不同的是每个数组元素会调用toLocaleString()方法转化为字符串，而不是调用各自的toString()方法.

Object.toJSON在待序列化的对象中存在这个方法，调用它，返回值即是序列化的结果

Object.valueOf()方法valueOf()和toString()类似，但往往当javascript需要将对象转换为某种原始值而非字符串时才会调用它，尤其是数字的时候。如果早需要使用原始值的上下文中使用了对象，javascript就会自动调用这个方法