在JavaScript中只有null和undefined值没有方法

JavaScript中所有数字均用浮点值表示，当一个数字直接出现在程序中时，称之为数字直接量

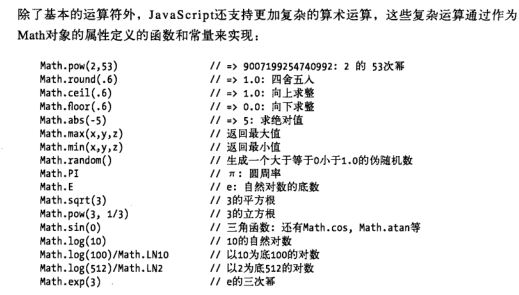
最好不要用以0为前缀的整型直接量，在ECMAscript6.0的严格模式下八进制直接量是禁止的

浮点型直接变量：除小数点外还可以用指数形式表示浮点型直接量，即在实数后加e或E，后面再跟正负号，这种方法是有前面的实数乘以10的指数幂

例：6.02e23 六点零二乘以十的二十三次幂

6.02E-23 六点零二乘以十的负二十三次幂

3.13 算术运算



JavaScript的算术运算在溢出，下溢或被零整除时不会报错，当结果超过JavaScript所能表示数字上限会以Infinity表示负无穷加符号，基于无穷大值进行的运算还是无穷大值

下溢是当运算结果无限接近于零比JavaScript能表现的最小数还小，这时会返回0

被零整除在JavaScript中不会报错，会返回无穷大值或负无穷大，还有一种情况，零除以零是没有意义的，运算结果是非数值，用NaN表示。无穷大除以无穷大，给负数开方或数字运算符与不是数字或无法转换为数字的操作数一起使用时都会返回NaN

NaN与任何数值都不相等包括自身，判断是否为NaN值不能用x==NaN，要用x！=x，当且仅当x为NaN时才会返回true

JavaScript中有两个函数，isNaN()如果参数是NaN或非数字值(比如对象和数组)时返回true，isFinite()在参数不是Nan，Infinity或-Infinity时返回ture

正零值和负零值是相等的，只在作为除数时不一样

3.2.1 字符串直接量

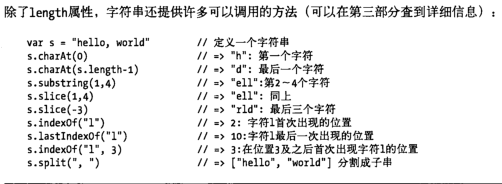
在JavaScript中的字符串直接量，是由单引号或双引号包括起来的字符序列，由单引号定界的字符串中可以包含双引号，反之亦然，在ECMAscript3.0中字符串必须写在一行而在ECMAscript5.0中字符串直接量可以拆成数行，每行必须以反斜杠(\)结束，如果希望在字符串直接量中另起一行可以用转义字符\n

最好在html和JavaScript中用不同的引号，html中用“”，JavaScript中用‘’

3.2.2 转义字符

\n换行\r回车\\反斜杠\f换页\”双引号\’单引号

3.2.3 字符串的使用



JavaScript中字符串是不可变的，类似方法返回的都是新字符串，原字符串本来没有改变

除了charAt()方法，还可以用[]来访问字符串中的单个字符

&&当且仅当两个操作数都为真时才返回true

||如果两个操作数其中一个为真就返回true

3.4 null和undefined

null是特殊的值类型是object，常来描述空值

Undefined的类型是undefined

两种值都没有任何属性和方法，用[]和.访问会产生一个类型错误，如果想作为参数传递还是用null好

3.5 全局对象

如果代码声明了一个全局变量，那么它就是全局对象的一个属性

3.6 包装对象

JavaScript对象是一种复合值：它是属性或已命名值的集合，通过”.”来访问，当属性值是一个函数时称为方法，类似o.m()调用

JavaScript会将字符串通过调用new String(s)的方法转换成对象，这个对象继承了字符串的方法，并被用来处理属性的引用，一旦引用结束就会销毁新创建的对象

var s = ‘test’;

s.len = 4;

vat t = s.len;

当运行着段代码时，t是undefined。第二行代码创建一个临时字符串对象，并给其len属性赋值为4，随即销毁这个临时对象，第三行代码通过没有修改过的字符串创建一个新的字符串对象尝试读取其len属性，值自然为undefined，修改只发生在临时对象上，存取字符串，数值，布尔值时创建的临时对象称作包装对象，且不能保存下来，字符串，数值，布尔值的对象都是只读的，且不能给他们定义新属性，需知它们与对象是有区别的，另可通过String()，Number()，Boolean()构造函数来显示创建包装对象

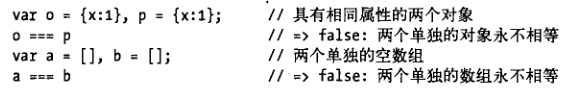
3.7不可改变的原始值和可变的对象引用

原始值number，string，布尔值，undefined，null不可变，对象，

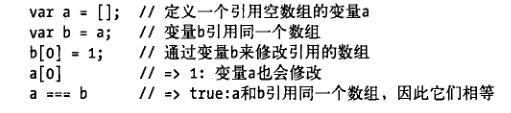
数组，函数都是可变的，字符串的方法都是返回一个新的字符串，

源字符串并没有变，number是换了个地址

原始值的比较是值的比较，而对象的比较并非值的比较，即使两个对象包含同样的属性同样的值也不相等



对象的比较均是引用的比较，当且仅当它们引用同一个基对象时，它们才相等，数组一样



将对象或数组赋值给一个变量，仅是引用，对象本身没有复制，想要复制必须显式复制每个属性或数组元素，同样，比较对象和数组也要分别比较其属性元素

3.8 类型转换

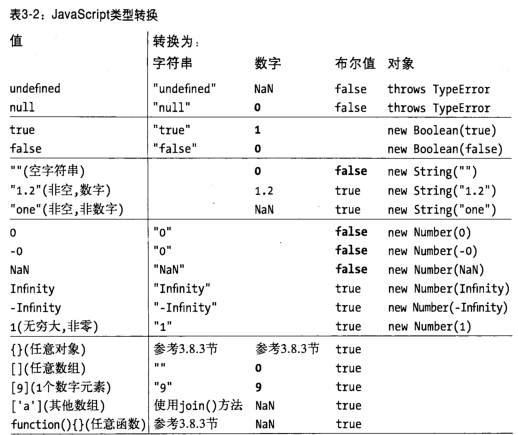
·10+”object” //==>”10object”数字10转化为字符串

·“7”\*”4” //==>28 两个字符串均转化为数字

·var n = 1 - ‘x’; //==>NaN：字符串无法转化为数字

·n + “object” //==>”NaNobject”:NaN转化为字符串NaN

以数字表示的字符串可以直接转换为数字，也允许在开始和结尾处有空格，但在开头和结尾有非空格的字符都不会被当成数字直接量的一部分，进而造成字符串转换为数字的结果为NaN



3.8.1 转换和相等性

“==”运算符很灵活，做运算时做了类型转换，”===”就不做

需要注意有时可以转换但不代表相等，在期望使用布尔值时undefined会转换为false，但undefined==false是错的，if语句将undefined转换为false

3.8.2 显示类型转换

JavaScript可以自动做很多类型转换，但有时仍需要做显示类型转换，为了使代码更清晰易读

显示类型转换用就是使用Object(),Number(),String(),Boolean()等系统自带构造函数来转换

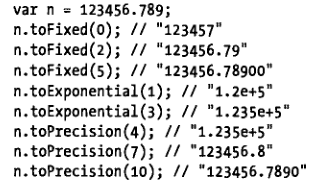
JavaScript提供了专门的函数和方法用来做更加精确的数字到字符串和字符串到数字的转换

Number类定义的toString()方法可以接收转换基数的参数，如不指定，默认为十进制，八进制：”O”+n.toString(8);十六进制：”OX”

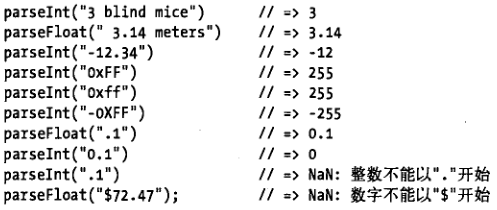
+n.toString(16)，二进制：n.toString(2)

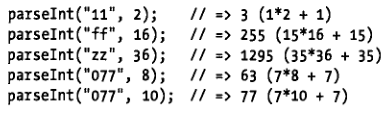
如希望控制小数点位置和有效数字位数，Number上定义了三个方法，toFixed()根据小数点后指定的位数将数字转换为字符串，它从不使用指数计数法，toExponential()使用指数计数法将数字转换为字符串，其中小数点前只有一位，小数点后的位数由参数决定(有效字位数比参数多一位)，toPrecision()根据指定的有效字位数将数字转换为字符串，如果有效字位数小于数字整数部分，

则转换成指数形式，三个方法都会四舍五入和适当添加0



parseInt()和parseFloat()(它们是全局函数不属于任何类的方法)parseFloat()则可以解析整数和浮点数，parseInt()只解析整数，它会跳过任意数量的前导空格，尽可能解析更多数字字符，并忽略后面内容，如果字符前缀是”OX”则视为十六进制数，如果第一个非空格字符是非法的数字直接量，将直接返回NaN，parseInt()可以接收第二个参数，这个参数指定数字转换的基数，在2~36之间





3.8.3 对象转换为原始值

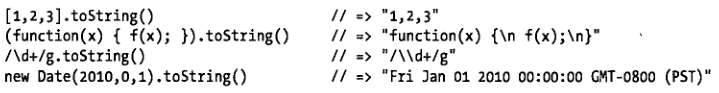
对象到布尔值都为true，包装对象也是：new Boolean(false)是一个对象不是原始值，它的布尔值也是true

对象到字符串和对象到数字是通过调用带转换对象的方法完成的，所有的对象继承了两个转换方法：

toString()，它的作用是返回一个反映这个对象的字符串，默认方法不会返回一个有趣的值

IMG_256

很多类定义了更多特定版本的toString()方法。例如。数组类，函数类，RegExp类，日期类：



另一个转换对象的函数是valueOf()默认方法只是简单地返回对象本身，而不是返回原始值。因为对象是复合值，而且大多数对象无法真正转换为原始值，数组,函数,正则表达式都是简单地继承了这个默认方法，日期类定义的valueOf()方法会返回一个内部表示：1970年1月1日以来的毫秒数

IMG_256

3.9 变量声明

var可以定义一个变量，可以定义多个变量，可以把变量赋值和变量声明放在一起写，如果未在变量声明中给变量赋值，变量的值就为undefined，JavaScript中变量可以是任意数据类型，例如

先赋值一个数字，再将字符串赋值给变量是合法的

重复的声明和遗漏的声明

在ECMAScript 5严格模式下不能给未声明的变量赋值，在非严格模式下可以给未声明的变量赋值，等同于全局变量，一定要避免

3.10 变量作用域

全局变量在全局作用域中，在函数内部声明的变量只在函数内部有定义，函数的参数也同样，在函数体内局部变量优先级高于全局变量，假如重名会覆盖

3.10.1 函数作用域和声明提前

变量在声明它的函数体以及这个嵌套的任意函数体内都是有定义的，嵌套结构上的父级访问不了子级定义的变量

函数里声明的所有变量都会被提前到函数顶部，函数声明整体提升变量声明提升，详见js笔记

3.10.2 作为属性的变量

当声明全局变量时相当定义了一个全局对象的属性，用var定义的全局变量不可用delete删除，但非严格模式下给未定义的变量赋值从而创建了一个全局变量，这个是可以用delete删除的

3.10.3 作用域链

详情见js笔记

表达式和运算符

4.1 原始表达式

最简单的表达式，是表达式最小单位，包含常量,直接量,关键字,变量，例如：1.23 “hello” /aaa/ true false this null

如果变量名不存在返回undefined，在ECMADcript 5严格模式下会抛出一个引用错误

4.2 数组和对象的初始化表达式

初始化表达式有时称数组直接量和对象直接量，数组表达式是由一对方括号和其内由逗号分割的列表构成的

IMG_256

这些表达式可以嵌套

IMG_256

JavaScript对数组初始化表达式进行求值时，数组初始表达式中的元素元素表达式也都会各自计算一次，数组初始化表达式每次计算的值可能是不同的

逗号之间省略的空位会填上undefined值

对象初始化表达式是由一对花括号和其中的由逗号分隔的子表达式构成，每个子表达式有属性名加冒号的前缀，对象直接量也可嵌套，对象直接量的属性名可以是字符串而不必是标示符(这在一些只能用保留字或非法标示符做属性名时非常有用)

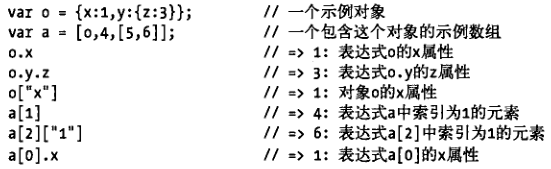
4.3 函数定义表达式

也可称为函数直接量var fn = function () {}

4.4 属性访问表达式

得到一个数组或对象的值，JavaScript定义了两个语法

IMG_256



会首先计算.和[]之前的表达式，如果是null或undefined会报错，如果不是对象会转换为对象，对象后是.则不用写””，如果是方括号中间要加上””，当属性名是保留字或者包含空格或标点符号又或者属性名是通过计算而不是固定的值时必须用方括号的写法

4.5 调用表达式

是调用函数或者方法的语法表示，如果函数用return返回一个值

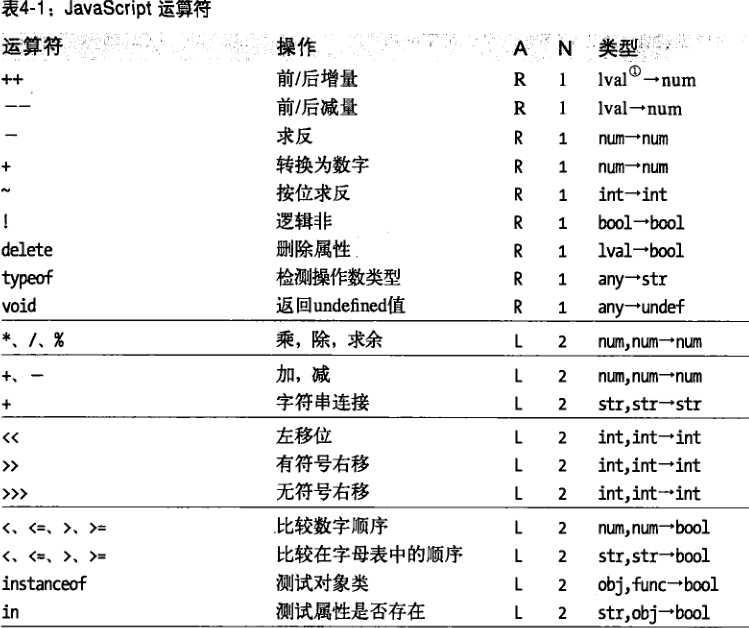
，那这个值就是整个调用表达式的值，如没有return一个值，则为undefined

4.6 对象创建表达式

创建一个对象并调用一个函数(被称为构造函数)，var a = new b()

4.7 运算符概述

标题为A，L(从左到右)R(从右到左)，标题为N，表示操作数的个数，标题为类型，操作数希望为什么类型



4.7.1 操作数的个数

大部分是二元运算符，有一元运算符，支持一个三元运算符，条件运算符”?：”，他将三个表达式合并成一个表达式

4.7.5 运算符优先级

表格顶部优先级最高依次向下越来越低

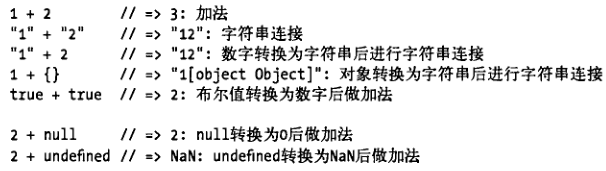
4.8 运算表达式

JavaScript中数字都是浮点型的，除数为0的结果为正无穷或负无穷，0/0结果为NaN，所以除法中，除数为0不会报错



同样，求余运算符也适用于浮点数，6.5%2.1结果是0.2

4.8.1 “+”运算符



运算结果也是依赖运算顺序的

IMG_256

4.9 关系表达式

4.9.1 相等和不等运算符

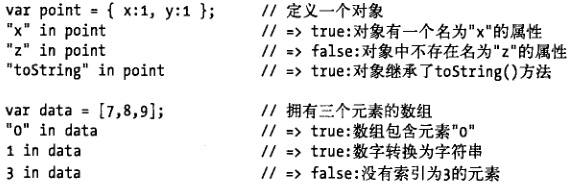
“==”经过类型转换，用X!==X来判断X是不是为NaN，undefined和null相等，自己互相也相等，如”1”==true，先将true转换为1，再将字符串1转换为数字1，”===”严格相等，先计算其操作数的值再比较，比较过程中没有任何类型转换，两个值类型不同不相等，null,Undefined,NaN都不等于他们自己，也不互相等

4.9.2 比较运算符

字符串比较是字符的数值进行比较是区分大小写的，所有大写ASCII码都小于小写的ASCII码，如果比较不需要区分大小写就先用String.toUpperCase()和String.toLowerCase()进行大小写转换，加号运算符偏向转换为字符串，而比较运算符偏向数字

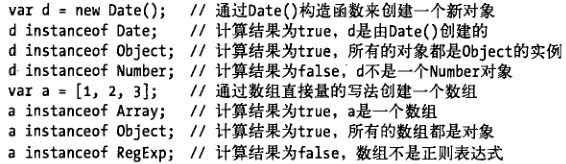
4.9.3 in运算符

In运算符希望它的左操作数是字符串或可以转换为字符串的，右操作数为对象，如果对象有名为左操作数的属性则返回true



4.9.4 instanceof运算符

instanceof运算符希望左操作符为对象，右操作数是一个函数，如果左操的对象是右侧函数的实例则返回true，否则返回false，如果左操作数不是对象则返回false，如果右操作数不是函数则会抛出错误，为了计算o instanceof f，先计算f.prototype，然后在原型链上找o，如果找到则返回true，o是f(或f的父类)的实例

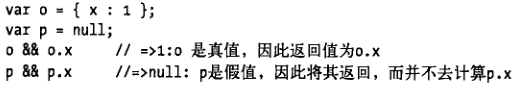


4.10 逻辑表达式

&& ,|| ,! 是对操作数进行布尔算术运算，常与关系表达式连用

4.10.1 逻辑与&&

操作数进行”与”操作，当两个操作数都为true时才返回true，关系运算符优先级比逻辑运算符高，如果左操作数为假值则不再继续计算右操作数，如果左操作数为真值，则计算右操作数的值并将其返回作为整个表达式的结果，隐式的return了



4.10.2 逻辑或||

操作数进行”或”操作，当两个操作数都为false时才返回false，它会首先计算左侧操作数，如为真则返回计算结果作为表示的结果，否则继续计算右侧操作数，并返回

4.10.3 逻辑非！

将操作数布尔值取反，它与||和&&不同，它先将操作数转换为布尔值，也就是说！总是返回true或false，而且！优先级很高

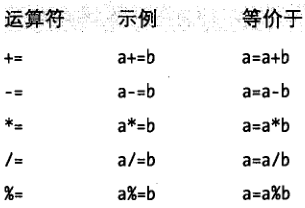
，与操作数绑定在一起，！&&=||，反之同样

4.11 赋值表达式

“=”具有非常低的优先级，赋值操作符结合性是自右向左，因此可以如下赋值：i=j=k=0，把三个标量初始化为0

带操作的赋值运算

“+=”：total += sales <=> total = total + sales



4.13 其他运算符

JavaScript支持各种各样的运算符

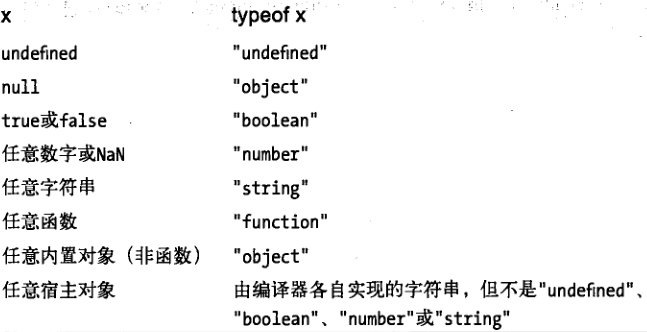
4.13.1 条件运算符 ?:

唯一一个三元运算符(有三个操作数)，第一个操作数在?之前，第二个在?和:之间，第三个在:之后，例如：x>0?x:-x 求x绝对值，

如果第一个操作数为真值则计算第二个操作数，如为假值则计算第三个操作数并返回作为结果，二三不可能同时计算

4.13.2 typeof运算符

是个一元运算符，放在操作数前面，返回值为表示操作数类型的字符串，表4-3列出了typeof返回值



typeof运算符可以带上圆括号，不管是对象还是数组都返回object，这对于区分对象和其他原始值是很有帮助得，真正的函数和可执行对象是有区别的，在ECMAScript 5中规定所有可执行的对象包括内置和宿主对象都返回function，在IE9以前非原生的可执行对象返回object

4.13.3 delete运算符

用来删除对象属性和数组元素，当它删除数组中元素时数组长度不变，通过var语句定义的变量不能删除，同样通过function定义的函数和函数参数也不能删除

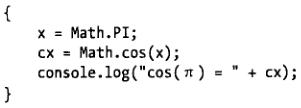
4.13.5 逗号运算符

二元运算符，先计算左操作数并忽略，再计算右操作数并返回

，只有左侧操作数具有副作用时，才会用逗号运算符

5.2 复合语句和空语句

可以用逗号表达式将几个表达式合在一起形成一个表达式，还可以用花括号将多条语句括起来形成复合语句



注意：第一，语句块末尾不加分号，但里面的语句加，第二，在语句块中声明的变量不是语句块私有

有时循环不需要语句或语句块，但规定循环必须有至少一条语句就用空语句代替，for() ;，在后面加个分号

5.3 声明语句

var语句和function()语句

5.3.1 var

5.3.2 function

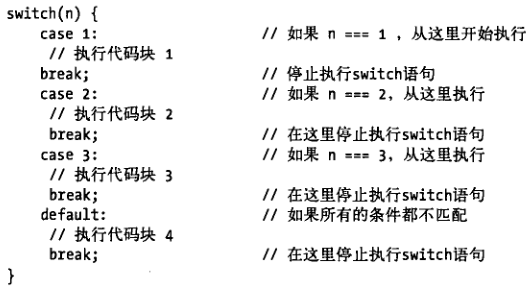
var f = function()函数定义 function f()函数声明

函数定义不能出现在if，for等其他语句中，var定义的只有有变量声明提前了，而使用函数声明语句函数名称和函数体均提前

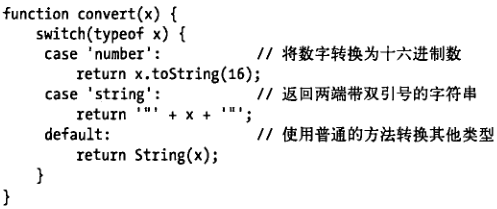
可以在一个函数声明前调用它，因为在上下文中都有，和var语句定义一样，函数声明也无法删除

5.4.1 if语句

5.4.3 switch语句



case和switch的表达式的值是否相同(这里的相同为”===”)，找不到匹配的case执行default，没有default它会跳过所有case语句，break可以跳出switch语句或循环语句，不加break会在匹配到合适的case后执行后面所有的代码，可以用return来代替break，case中最好写常量表达式



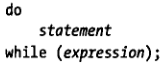
5.5 循环

5.5.1 while循环

while(expression) statement 首先计算expression，如为真值执行循环，继续计算expression，直到为假值退出循环

5.5.2 do while

它在尾部检查循环表达式，意味着循环体至少执行一次



且do while得用分号结尾，while(true)死循环

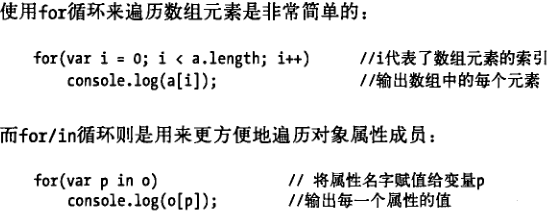
5.5.3 for循环

for(initiolize; test; increment)初始化，检测，更新

for循环中任意一个表达式可以省略，但分号不能省略，如果忽略检测就是死循环，for(;;)就是死循环

5.5.4 for in循环

与for完全不同，for(variable in object) statement，variable通常是变量名，产生左值的表达式或用var定义的变量，object是一个表达式，这个表达式的结果是对象



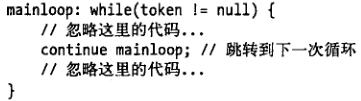
只有可枚举的属性才会被遍历到，语言内置定义的方法和很多属性就不可被枚举，自定义的属性和方法可以被枚举，包括从其他对象继承到的，for in循环中被删除的还未枚举到属性将不会再被枚举到，每次循环会重新计算variable表达式，例如将对象属性名赋值到数组中

var a = {name:””, age:””} var b = [], i = 0; for(b[i++] in a);

5.6 跳转

break语句可以跳出循环，continue结束本次循环进入下一次循环，return跳出函数体的执行，并返回本次执行结果，throw是与try/catch finally语句一同使用

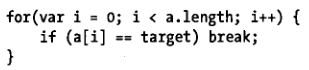
5.6.1 标签语句



Break和continue是JavaScript中唯一可以使用便签语句的语句

5.6.2 break语句

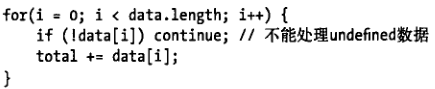
单独使用是跳出最内层的循环或跳出switch循环，当结束条件复杂的时候，可以找到目标后直接break，不必写复杂的结束语句



break lablename；终止这个闭合语句块的执行，这个带标签的语句不应该是循环或switch语句，可以是单独一个语句块，希望用break跳出非就近的循环体就要用到便签语句了

5.6.3 continue语句

使用方法与break一样，也可用标签语句，结束本次循环执行下一次循环，不管带不带标签只能出现在循环体内



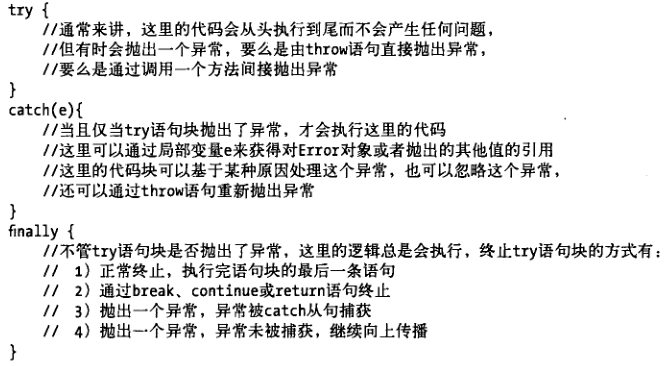
5.6.4 return语句

只能出现在函数体内

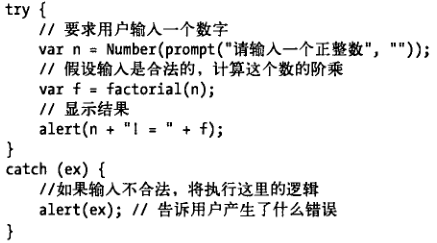
5.6.5 throw语句

抛出异常，语法：throw expression；expression可以是任意类型，代表错误的数字，或者包含可读的错误信息的字符串，Error对象有一个name属性表示错误类型，，message属性用来存放传递给构造函数的字符串throw new Error(“xxxxx”)，当抛出一个错误，会跳转到最近的异常处理程序，有try/catch/finally的catch从句编写，如果抛出异常的代码块没有处理程序就向上找，找不到就客户端报错

5.6.6 try/catch/finally语句



catch标示符有块级作用域，只在catch中有定义，例子：factorial()阶乘，prompt()和alert()输入输出，finally不常用



5.7 其他语句类型

width，debugger，和use strict

5.7.1 with语句

临时增加原型链，不推荐用，维护很麻烦，效率低

5.7.2 debugger语句

这条语句产生一个断点，使代码执行停在这一句，再通过调试器检测错误，必须调试器在运行中才好使，debugger自己不会启动调试器 语法xxxx debugger；

5.7.3 “use strict”

只能出现在脚本代码的开始和函数体的开始，意思是后续代码会是严格模式下的，严格模式有几种：

·不能用with语句

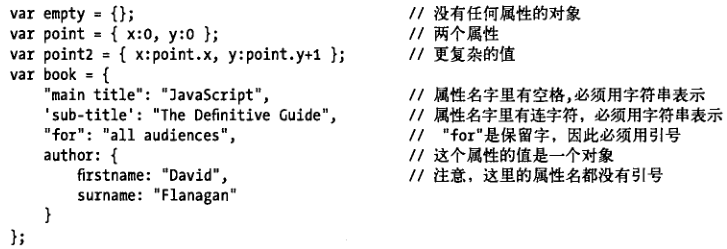
·所有变量都须先声明

1. 对象

除了原始值之外都是对象，对象常用的方法是创建，设置，查找，删除，检测和枚举它的属性，内置对象，宿主对象，自定义对象，自由属性，继承属性

6.1 创建对象

6.1.1 对象直接量



对象直接量是表达式，如果在一个重复调用的函数中的循环体内使用对象直接量可能会导致每次创建的不一样

6.1.2 通过new来创建对象

var o = new Object();

6.1.3 原型

通过new和构造函数创建的对象的原型就是构造函数的prototype属性的值，没有原型的对象不多，Object.prototype就是其中之一，所有内置构造函数及大部分自定义构造函数都有继承自Object.prototype的原型

6.1.4 Obiect.create()

6.2 查询

用方括号访问时必须是字符串或可以转换为字符串的值

6.2.1 作为关联数组的对象

在预先不知道对象属性名时，想要访问这个属性就要设变量，用[]来访问，因为[]中传的是字符串是动态的，而.是写标示符，标示符是写死的，用for in循环

6.2.2 继承

假如要查询对象o的属性x，o中没有就找它的原型，直到为null

，这就构成了原型链，如果之前继承了同名的属性，再写一个会覆盖掉继承的，只有在查询时才会体会到继承，设置不会影响到原型链

6.2.3 属性访问错误

如果不确定对象的属性是不是对象，例book.subtitle.length，如果book和book.subtitle不全是对象book.subtitle.length就会报错，var len = book && book.subtitle && book.subtitle.length解决

内置构造函数是只读的，在严格模式下设置它会报错

6.3 删除属性

delete book.author;delete只能删除自有属性，不能删除继承属性，要遍历属性的属性，依次删除，以免造成内存泄漏

6.4 检测属性

In “x” in o; o.hasOwnProperty(“x”); in继承属性有也算，hasOwnProperty只有自有属性才算

propertyIsEnumerable()只有检测到属性可被枚举才会返回true

6.5 枚举属性

有时数据库给Object.prototype添加了许多属性和方法，不能定义为不可枚举，所以要过滤

for(var p in o){if(!o.hasOwnProperty(p)) continue;} //跳过继承属性

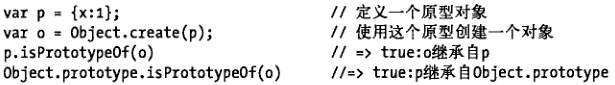
for(var p in o){if(typeof o[p] === “function”) continue;} //跳过继承方法

6.8 对象的三个属性

每一个对象都有与之相关的原型，类和可扩展性

6.8.1 原型属性

对象的原型属性是用来继承属性的，经常会把”o的原型属性”叫做”o的原型” ，对象直接量创建的对象使用Object.prototype作为它的原型，将对象作为参数传到Object.getPrototypeOf()可以查询它的原型，p.isPrototypeOf(o);检查p是否是o的原型，与instanceof运算符类似



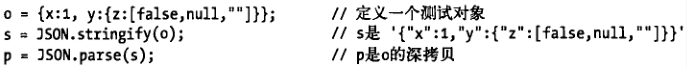
6.8.2 可扩展性

内置的，自定义的都可扩展，可用Object.preventExtensions()，将待转换的对象传进去变为不可扩展的，只影响本身不影响原型，Object.seal()将对象也变为不可删除，Object.freeze()再将对象属性数组变为只读

6.9 序列化对象

对象序列化是指将对象转换为字符串并可以转换回来，JSON.strin

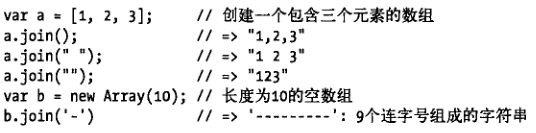
gify()和JSON.parse()



JSON.stringify(o)只能序列化对象的可枚举属性

1. 数组

7.8.1 join



7.8.2 reverse

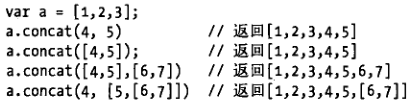
IMG_256

7.8.3 sort

不传参数按字母表顺序排，传比较函数，返回负的第一个参数在前，返回整的，第一个参数在后

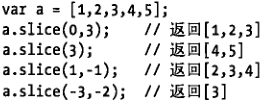
7.8.4 concat

创建并返回一个新数组，不会修改原数组



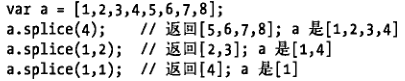
7.8.5 slice()

返回一个子数组，两个参数分别指明开始和结束的位置，但不包含第二个参数，传负数是倒过来，不会修改原数组，如只有一个参数包括参数往后所有元素，-1指定倒数第一个元素



7.8.6 splice

插入或删除的通用方法，会改变原数组，会改变索引值，第一个参数指定了插入或删除的位置，第二个参数是删除的个数，省略就是从开始到结尾全部删除，把删除的元素变成新数组返回，如果没删除元素就返回空数组

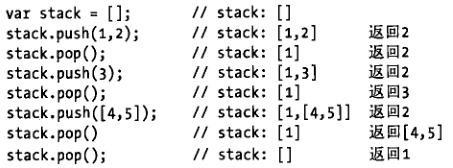


紧随其后的任意参数个数指定了需要插入到数组中的元素，从第一个参数指定的位置开始插

IMG_256

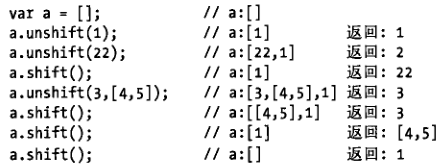
7.8.7 push和pop

push()在数组尾部添加一个或多个元素，pop()删除最后一个元素，并返回删除的值，修改调用的函数



7.8.8 unshift()和shift()

unshift()在数组头部添加一个或多个元素，shift()删除第一个元素，并返回删除的值，修改调用的函数



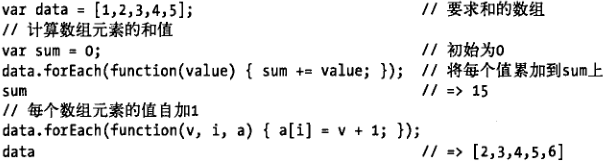
多个参数一起unshift添加的顺序与参数顺序一致，分别插入顺序相反

7.9 ECMAScript 5数组方法

一般都有一个函数，函数可以传三个参数：数组元素，数组元素索引，数组本身，但可以忽略两个

7.9.1 forEach()

从头到尾遍历数组，为每个元素调用指定的函数



但forEach无法中途停下，for循环可以用break停，不改变a

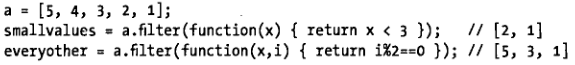
7.9.2 map()

从头到尾遍历数组，为每个元素调用指定的函数，并返回一个数组，它包含该函数的返回值，不改变a

IMG_256

7.9.3 filter

返回的是调用的数组的一个子集，从头到尾遍历数组，传递的函数进行逻辑判断，判定为true的元素就是这个子集的成员



7.9.4 every和some

是数组的逻辑判断，返回true或false，对传递的函数进行逻辑判断，every相当于任意，当且仅当调用数组所有元素都返回true才返回true，some相当于存在，至少有一个为true则为true，

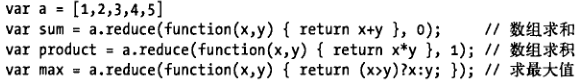
判断false时every是存在，some是任意

IMG_256

IMG_256

7.9.5 reduce和reduceRight

使用指定的方法将数组元素组合形成单个值

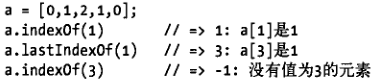


reduce需要两个参数，第一个是要调用的函数，将两个值化简成一个，第二个(可选)的参数为传递给函数的初始值，x是0，y是第一个数组元素，经计算后计算结果给x继续计算，第二个参数可省略，省略掉后第一二个参数就是数组第一二个元素，不指定第二个参数的空数组会报错

reduceRight就是从右侧开始化简

7.9.6 indexOf和lastIndexOf

搜索整个数组中的特定值，返回找到的第一个元素的索引，没有返回-1，lastIndexOf从最后开始搜索，无法找到NaN，判断方式===



第一个参数不接受函数要传搜索的值，第二个参数可选，传一个索引值，从那开始找，字符串也有indexOf和lastIndexOf

7.10 数组类型

array.isArray();用来判断是否为数组，在ECMAScript 5之前很难判断，instanceof不一定准

7.11 类数组对象

7.12 作为数组的字符串



数组的一些方法可以用到字符串上，但字符串是不可变的值，会修改数组的方法放在字符串上都不无效

1. 函数

8.1 函数定义

以函数表达式方法定义的函数是无法在未被定义前调用的，函数声明不能写到循环，条件判断等其他语句中，但函数定义表达式可以，它可以出现在代码任何地方

8.2 函数调用

作为函数，作为方法，作为构造函数，用call()和apply()方法间接调用

8.3 函数的形参和实参

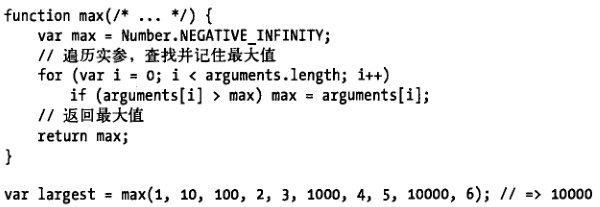
当调用函数时的实参个数和声明时的形参个数不符时的状况，如何显示地检查函数实参类型，以免非法的实参传进来

8.3.1 可选形参

当实参比形参少，剩下的形参为undefined值，假如要传一个数组a，可以写var a = a || []，这样没传a也没关系

8.3.2 可变长的实参列表：实参对象

在函数体内实参是一个类数组对象，由arguments引用，可以用arguments[]来调用实参，以数字为索引，arguments也有length属性，通常不必检查实参个数，省略的实参将是undefined值，多出来的会被忽略，实参对象有一个重要作用就是让函数可以操作任意个数的实参



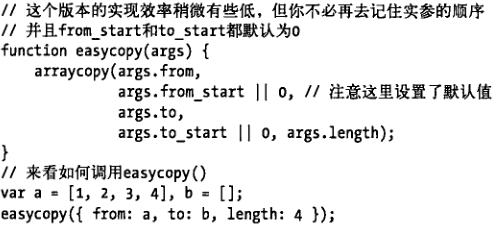
这种函数被称为不定实参函数，实参名arguments[0]和第一个形参相当于同名变量，改一个另一个也会变，在ECMAScript 5中移除了实参对象的这个特殊特性

caller和callee

实参对象还定义了这两个属性，在ECMAScript 5严格模式下会出错，非严格模式下有时很有用，arguments.callee(实参)可以调用函数自身

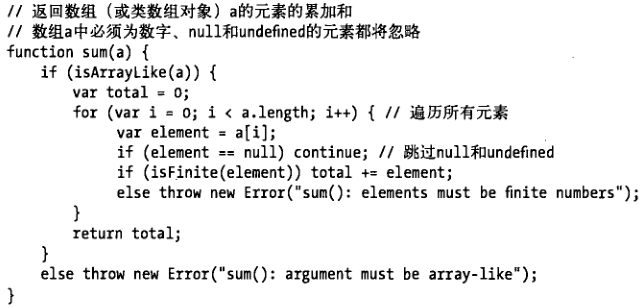
8.3.3 将对象属性用作实参

形参太多记不住顺序，传对象，用名-值的形式来传参数，传入的实参都写入一个对象中，在调用的时候传入一个对象



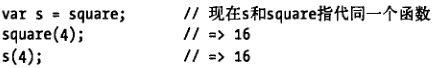
8.3.4 实参类型

为了避免传入实参类型不符造成的bug，需要类型检测机制



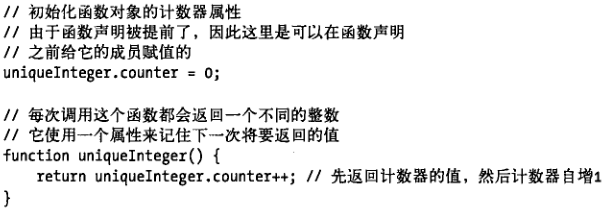
8.4 作为值的函数

可以将函数赋值给变量，存储在对象的属性，数组的元素中，作为参数传给另一个函数



8.5 自定义函数属性

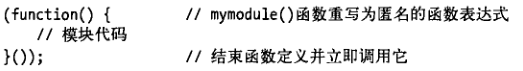
函数是一个特殊对象，可以拥有属性，当函数需要一个”静态变量”来保持调用时某个值不变，最方便就是加函数的属性，定义全局变量是很差的方法，因为只有这一个函数需要这个值



8.5 作为命名空间的函数

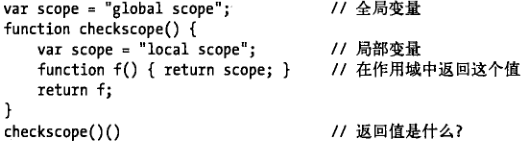
如果要得到一个储存计算结果的变量，但到不同程序中无法得知该变量是否被定义，如果新程序存在这个变量那么将会和代码发生冲突，解决办法是将代码块放到函数内，在调用得到返回值

可以直接定义一个匿名函数，在单个表达式中调用



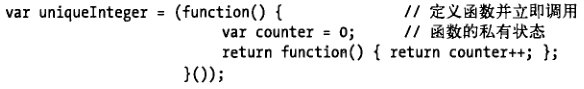
8.6 闭包

当一个函数嵌套另外一个函数，外部函数将嵌套函数作为返回值返回时往往会发生这种事情



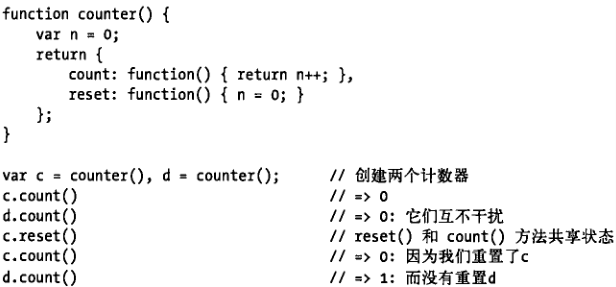
返回值是local scope，return f将checkscope执行，f定义时候的作用域链保存出来，不管在哪调用f，f调用的变量都是原先作用域链的，就像返回f同时返回了以前f定义时的作用域链

私有变量：



立即执行函数关注返回值，其实是把返回的嵌套函数保存起来，这样counter就只有被保存出来的函数才可以访问，实现了私有变量

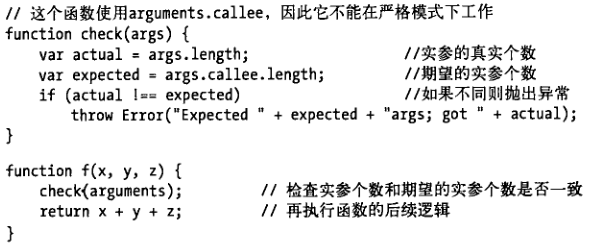
在同一个外部函数内定义的嵌套函数都共享一套作用域链



因为每次执行counter都重新创建作用域链，所以它们互不干扰

8.7 函数属性和方法，构造函数

8.7.1 length属性



8.7.2 prototype属性

每一个函数都包含一个prototype属性，这属性是指向一个对象的引用，这个对象称作原型对象，当函数用作构造函数时，新创建对象都会从原型对象上继承属性

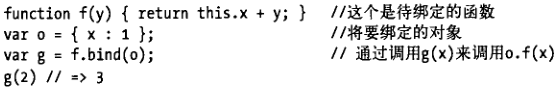
8.7.3 call()和apply()

第一个参数是要调用函数的母对象，要想以对象o的方法来调用函数f()，f.call(o) f.apply(o)，第二个参数传实参call直接传，apply以数组形式传，apply第二个参数可以是类数组对象

IMG_256IMG_256

8.7.4 bind()

函数f调用bind，绑定到一个对象上并返回一个新函数，作为它的方法，传实参传到新函数中



8.7.5 toString()

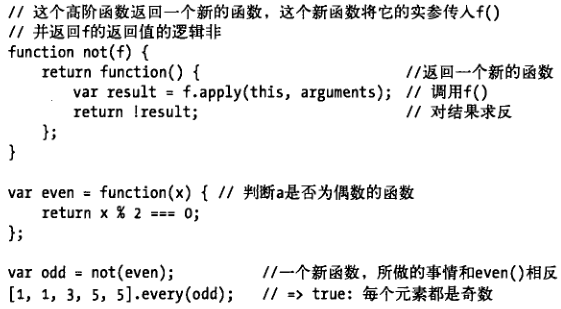
自定义的返回完整源码，内置的返回[native code]

8.7.7 可调用的对象

8.8 函数式编程

8.8.2 高阶函数

操作函数的函数，将函数作为实参传入并返回一个函数



8.8.3 记忆

1. 类和模块

每个JavaScript对象都是一个属性集合，相互没有任何联系，但JavaScript中也可以定义类，让每个对象都共享某些属性，类的实现基于原型继承机制

9.9 模块

9.9.1 用作命名空间的对象

用对象作为命名空间，变量和函数为对象的属性

1. 正则表达式的模式匹配

RegExp表示正则，String和RegExp都定义了方法

10.1正则表达式的定义

可以用new调用构造函数RegExp()来创建对象，更多的是直接量

var reg = /s$/; 以s为结尾的字符串