**深入理解JavaScript 第一二部分**

**第1部分 JavaScript快速上手**

1. 背景
   * 1. JavaScript与ECMAScript

JavaScript指的是一种编程语言。

ECMAScript是这种语言规范的名字。

* + 1. JavaScript的影响和本质

JavaScript的创造者：Brendan Eich

1. 语法

1.2.1 语法概述

等于号的两种不同用法：

• 一个单独的等于号(=)， 用于为变量赋值。

• 三个连续的等于号(===)， 用于比较两个值

1.2.2 语句和表达式

1. 语句“做事情”，程序是一系列的语句的集合
2. 表达式产生值，通常是函数的参数，或者是赋值的右边部分。

var x = y >= 0 ？ y ：-y ；（可以用于函数的参数，而语句不行）

myFunction（ y >=0 ? y ：-y ）（做函数的参数）

1. 表达式可以用在所有需要语句的地方。

1.2.3 分号

分号用于结束语句，而不是语句块。

1.2.4 注释

单行注释 //

多行注释 /\* \*/

1. 变量和赋值

1.3.1 赋值

变量在声明后使用。

变量声明和复制可以同时进行 var foo=6; 也可以为一个已有变量进行赋值 foo=4;

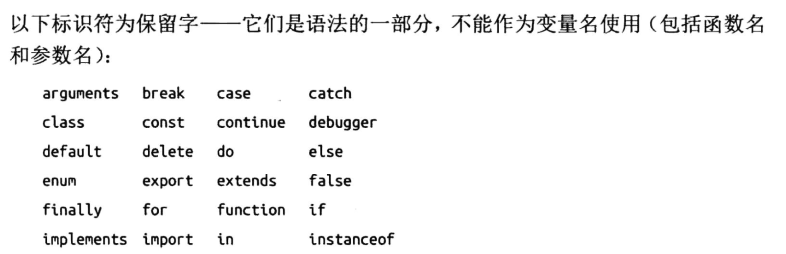
1.3.2 复合赋值运算符

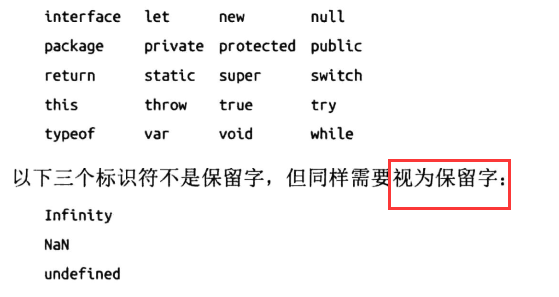
如：+= x+=1； 等同于 x = x+1；

1.3.3 标识符与变量名

• 标识符是JavaScript中各种语法的名称。（变量的名字就是一个标识符，标识符区分大小写）

* 标识符的第一个字符可以是任意的字母、＄，下划线 \_ , 后面的字符可以是数字。





1. 值

所有的值都有属性，每一个属性都有一个key（或者是name）和一个value。通过点（· ）操作符可以读取属性或者给属性赋值。

1.4.1 原始值和对象

• 原始值包括布尔值、数字、字符串、null和undefined。

• 其他的值都是对象。

• 区别：比较方式不同，每个对象都有唯一的标识且只（严格地）等于自己。而所有的原始值只要编码值相同，则被认为相等。

1.4.2 原始值

• 布尔值：true，false

• 数字：1，0，1.34

• 字符串：“abc” 、“hello”

• 两个“空值”：undefined ， null

• 原始值的特点：

1. 按值进行比较

内容比较：3===3 （类型和值都相等）→ true

1. 不可改变

其属性不能被改变、添加或者移除

1.4.3 对象

• 所有的非原始值都是对象

* 简单对象：可以通过对象字面量来创建
* 数组：可以通过数组字面量来创建
* 正则表达式：可以通过正则表达式字面量来创建 /^ a + b +$/
* 对象的特点：

1. 按引用进行比较

比较身份标识，每个值都有各自的身份标识 { } === { } → false

1. 默认可变

对象属性可以很自由地被改变、添加和移除

1.4.4 undefined 和null

• undefined表示“没有值”，未被初始化的变量为undefined ；访问不存在的属性，得到undefined ，丢失的参数是undefined

• null表示“没有对象”，在用到对象的时候它表示空值（例如参数、对象链中的最后一个元素等）

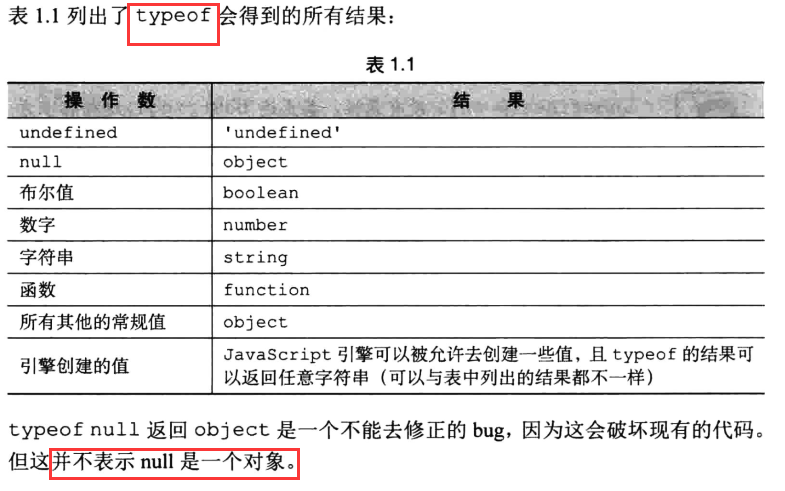
• 检查undefined 或null

函数允许通过undefined 或null 表示缺失的值，可以通过显式的检查做到同样的事 if ( x===undefined || x===null) ,也可以用将undefined 和null都可被视为false来处理 if ( ! x )

false , 0 , NaN, “ ”都可被视为false

1.4.5 使用typeof 和 instanceof 对值分类

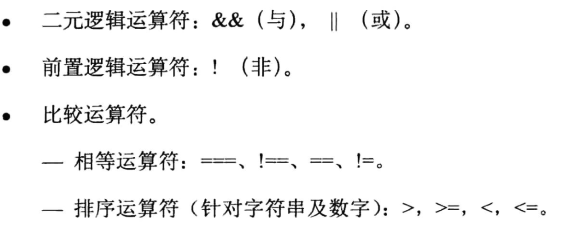
对值进行分类的操作符：typeof主要用于原始值 ；instanceof用于对象。





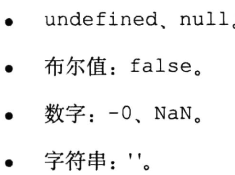
1. 布尔值

原始布尔类型包含 true 和false 两个值，以下运算符会产生布尔值



1.5.1 真值与假值

以下的值会被解释为false



其他所有值（包括所有的对象）都会被当成true，称为真值，被称为false的值为假值。

1.5.2 二元逻辑运算符

1）与（&&）

如果第一个运算数是假值，返回他；否则，返回第二个运算数。

123 &&“abc” → “abc”

2）或（ | | ）

如果第一个运算数是真值，返回它；否则，返回第二个运算数。

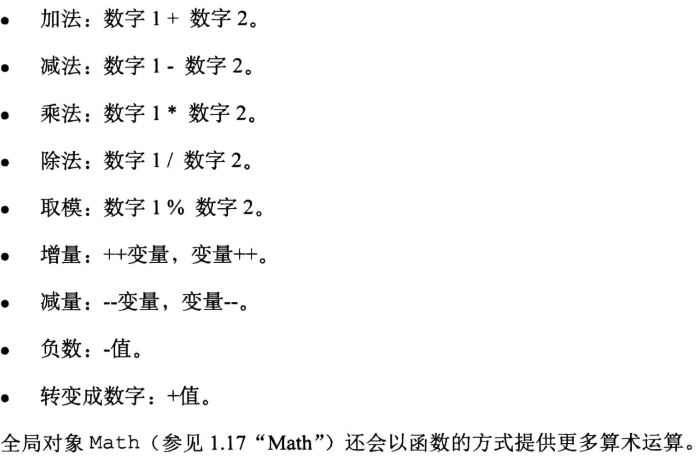
‘ ’ || 123 → 123

1.5.3等式运算符

常规等式：== 或 != (值相等)

严格的等式：=== 或 !== （值和类型相等）

1. 数字
2. JavaScript中所有的数字都是浮点数。 1===1.0 → true
3. NaN( not a number) 一个错误的值
4. Infinity 多数情况下也是一个错误的值。 3/0 → Infinity
5. Infinity比任意一个数都要大（NaN除外），- Infinity比任何一个属都要小（NaN除外），当需要一个最大值和最小值时将这两个数字作为默认值。
6. 运算符（算术运算符，位运算符（“与“，”或“，”非“））



1. 字符串

• 字符串可以直接通过字符串字面量来创建，（’ ’ ,” ”）( \ )反斜杠用来转义字符及产生一些控制字符。

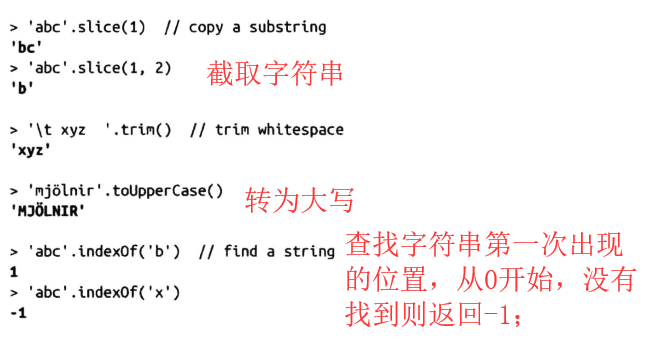
• 字符串的length属性可以对字符的个数进行计数。字符型是不可变的。

1.8.1 字符串运算符

• 字符串可以通过加号(+)进行连接，如果其中一个运算数是字符串的话，另一个运算数将被转换为字符串:

• 要在多个步骤中连接字符串，可以使用+=运算符:

1.8.2字符串方法



1. 语句

1.9.1条件语句

If……else语句；switch（）{ case ： }语句

1.9.2 循环语句

for（）语句； while（）语句 ； do……while语句 （break跳离循环，continue开始新循环迭代）

1. 函数

通过函数声明的方式来定义函数，也可以将函数作为直接参数传递给另外的函数

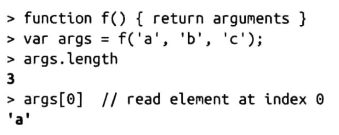
1.10.1 函数声明的提高特性

函数声明具有提高特性，var声明也具有提高的特性（但是通过其执行的赋值不具备该特性）

1.10.2特殊的变量 arguments

函数的所有参数都可以被自由调用，它会通过arguments变量来使所有参数可用。

arguments看起来像个数组，但却不具备数组的方法。

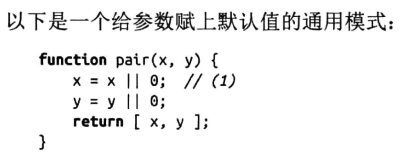


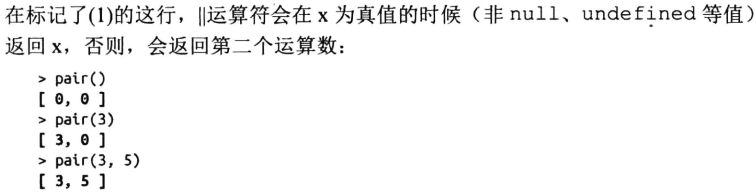
1.10.3参数太多或太少

额外的参数会被忽略

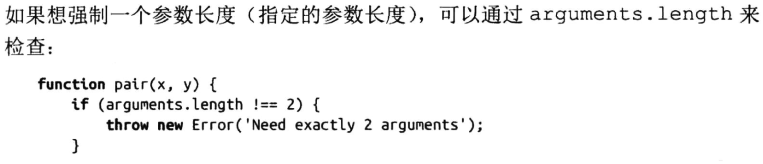
丢失的参数会得到undefined这个值。

1.10.4可选参数

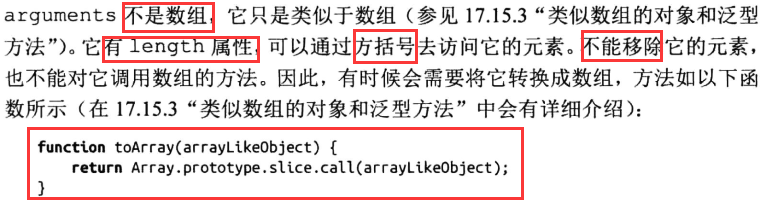




1.10.5 强制参数长度



1.10.6将arguments转换为数组



1. 异常捕获

使用try语句包裹关键代码，如果try语句有异常会被抛出那么catch语句就会执行。

1. 严格模式

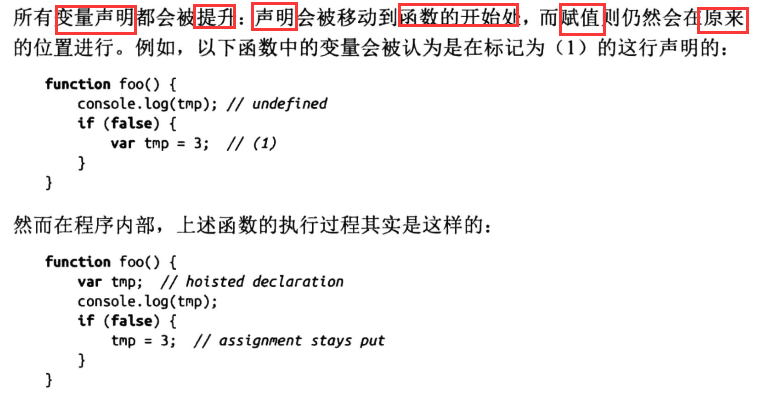
严格模式激活更多的警告以及使JavaScript变得更加干净，也可以在函数里面激活它。

1. 变量作用域和闭包

1.13.1 变量是函数作用域的

一个变量的作用域总是完整的函数（相当于前块）

1.13.2 变量的提升特性



1.13.3 闭包

每个函数都和它周围的变量保持着连接，哪怕它离开被创建时的作用域也是如此。

函数以及它所连接的周围作用域中的变量即为闭包。

1.13.4 IIFE模式:引入一个新的作用域

立即调用函数表达式

IIFE用例：闭包造成的无意共享

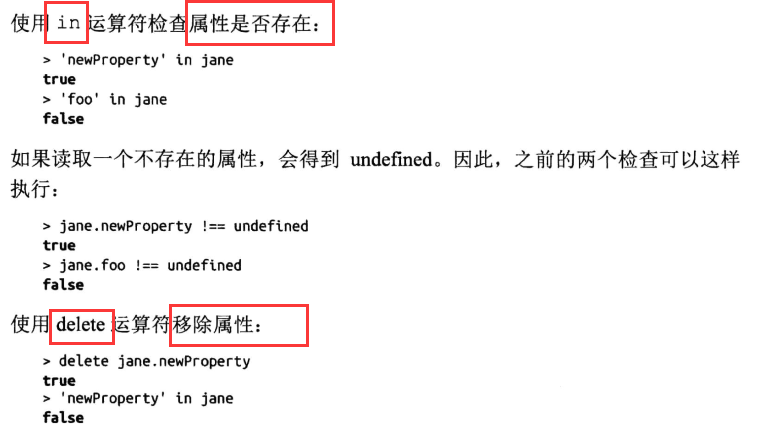
1. 对象和构造函数

1.14.1 单一对象

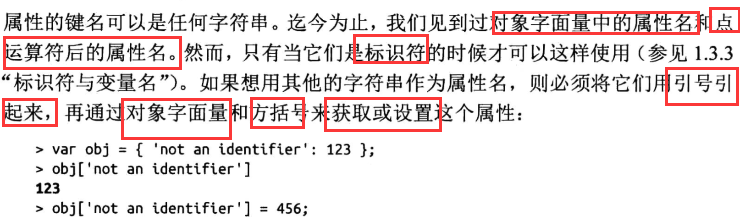
对象具有属性。对象是一组属性的集合，每个属性都是一个(键，值)对。键名都是 字符串，而值可以是JavaScript的任意值。

在JS中，可以直接通过对象字面量去创建普通对象。

以函数作为值的属性被称为方法



1.14.2 任意属性名



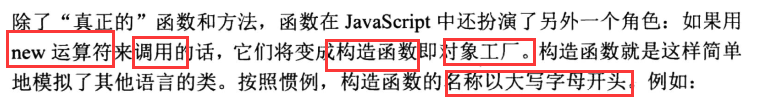
1.14.3 提取方法

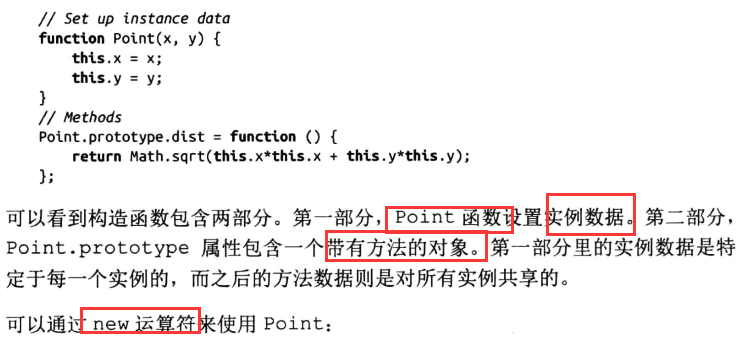
如果对方法进行提取，则会失去与对象的连接。处理这个问题的解决方案可以使用bind()方法，所有函数都支持。它会创建一个this总是指向给定值的新函数。

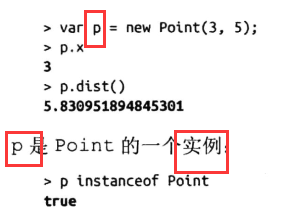
1.14.4 方法中的函数

在嵌套函数内部不能访问方法中的this变量。解决：可以将this保存在不同的变量中，或者利用参数给this指定一个值。？

1.14.5 构造函数：对象工厂



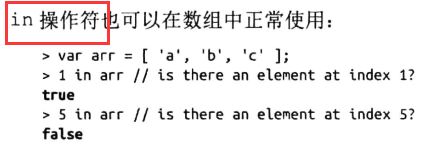




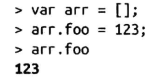
1. 数组

1.15.1 数组字面量

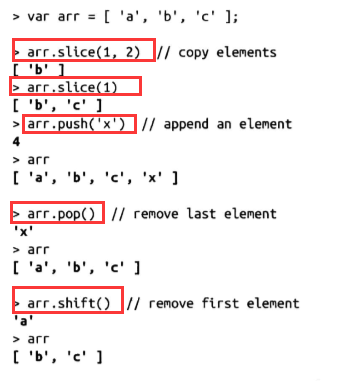
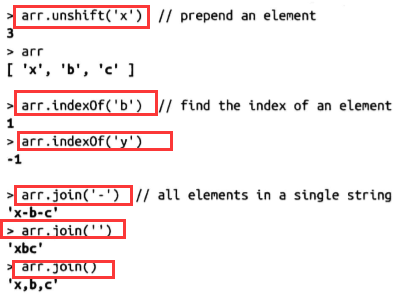
length属性表明数组有多少元素。你可以通过使用它来添加或者删除元素:



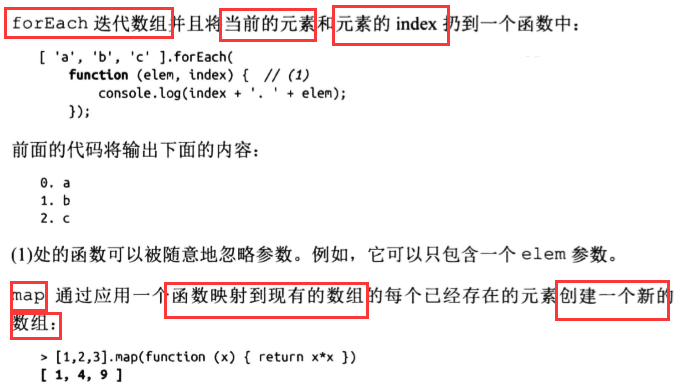
数组是对象，可以拥有对象属性



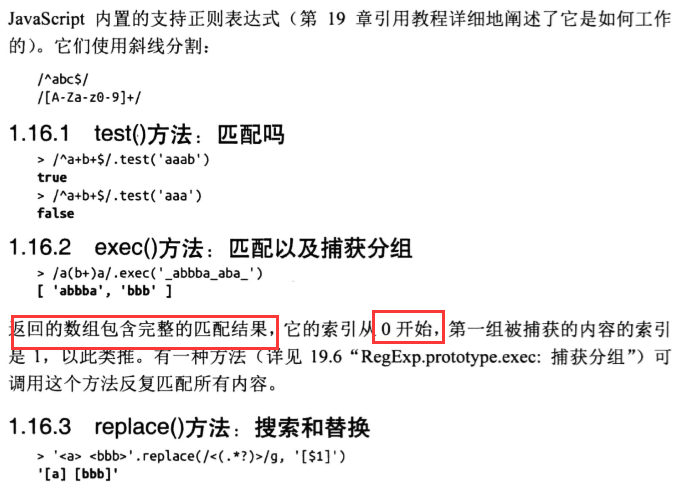
1.15.2 数组方法

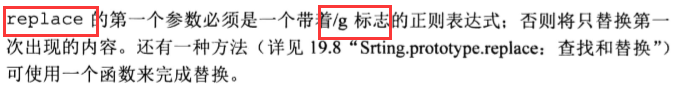
 

1.15.3 遍历数组

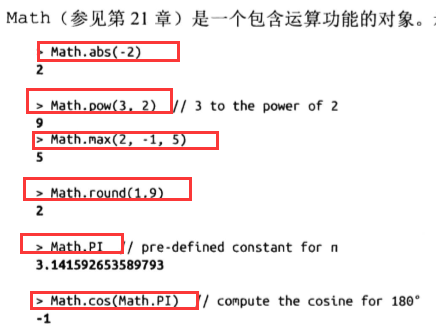


1. 正则表达式





1. Math



第2部分 背景

第三章 JavaScript性质

1. 他是动态的，是动态类型（变量和属性值可以保存任意类型。），他是函数式和面向对象的，它静默失败（最初不抛出异常），它部署的是开源代码，

2. 优雅部分

