第一章 让自己习惯js

第一条：为了达到更好的普遍性，建议在严格模式下编写代码。

不严格的js连接在严格的js后面，会被视为严格的js，所以严格的js可以写在自调用函数内。

(function(){

‘use strict’

........

})()

1. ：js中只有双浮点数。

位运算符会做一些多余的操作，但是优化编辑器有时候会推断出这些情形而在内部将数字以整数方式存储以避免多余的转换。

0.1 + 0.2 = 0.3000000000000004,所以浮点数运算时可以将其转为为整数再计算，如(1+2) / 10。

1. ：：数字与true进行运算时，将true强制转化为1，与false和null运算时强制转化0，与underfined运算时，结果为NaN。

位运算符会将字符串强制转化成数字。

+进行字符串和数字运算时，会转化为字符串，-\*/会将字符串转化为数字。

‘j’ + {toString: function () {return ‘s’;} }; //”js”,对象通过隐式得调用其自身的toString方法

2 \* {valueOf: function () { return 3; } }; //6,类似上条

而当对象同时拥有valueOf和toString时，js优先取valueOf。

如果对象有一个valueOf产生数字，最好有一个toString方法产生数字的字符串。

强制装换为false的值，0，undefined，null，’’，NaN。

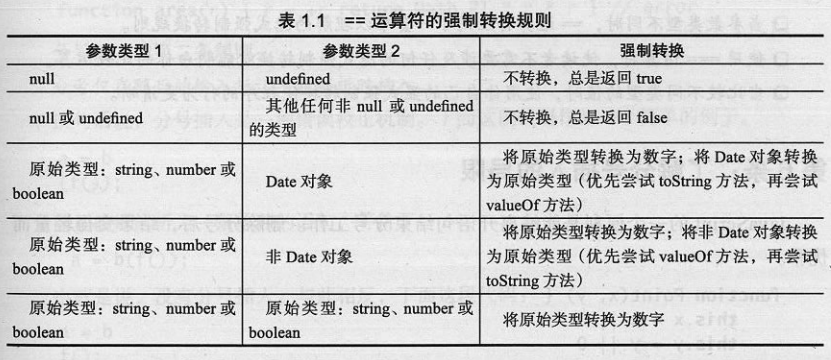
1. ：原始类型，布尔值，数字，字符串，undefined，null。

封装对象存在的意义只在于它们的实用方法。

1. ：==运算符，会进行强制转换，结果类似第三条，如果对象存在valueOf或者toString的方法，会调用这些方法。

使用==便会使以后读代码的人强制去记住这之中的强制转换。

==视图将对象转换为原始值，Date对象调用方法的顺序相反，优先尝试调用toString。



1. ：分号插入第一条规则：只能在一行、一个代码块和一段程序结束的地方省略分号。

第二条规则：分号仅在随后输入标记不能解析时自动插入。

a = b 等价于 a = b(f()) b解析为函数所以不会自动加入分号。

(f())

如果以( 、[、+、-、\*、/字符开头的代码可能不会加入自动插入的分号。

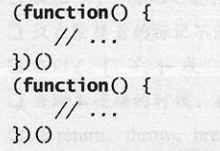
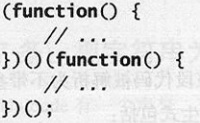
a = b 等价于a = b[‘r’,’g’,’b’]

[‘r’,’g’,’b’].forEach(function (key) {})

B[‘r’,’g’,’b’]等价于b[(‘r’,’g’,’b’)]而(‘r’,’g’,’b’)返回b所以就等价于b[‘b’]。

/要注意以正则开头的代码，会被解析成除号。

由第一条所说，严格代码要写在自运行函数内，而文件之间连接的时候就有可能导致



为了确保安全，文件开头以上述六个字符开头的都应该在开头加上分号。像

;(funciton(){})()

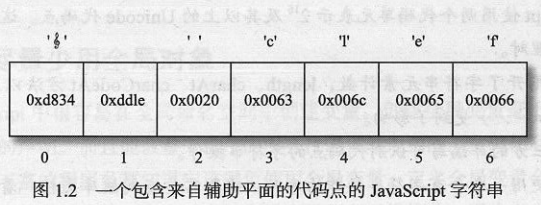
在自增自减前面会自动添加分号。

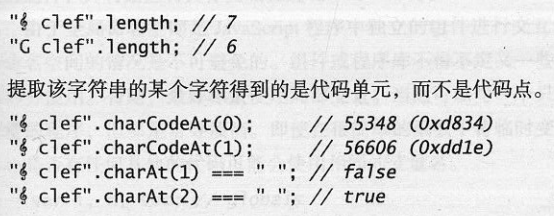
A

++ 解析为 a; ++b;

B

第三条规则：在for循环定义时，不会自动添加分号，在while空语句时，应该手动添加分号。

1. ：在utf-16编码下，





Js使用两个代码单元表示2的16次方及其以上的unicode代码点。这两个代码单元被称为代理对，代理对甩开了字符串元素计数，length、charAt等方法以及正则收到影响。