**JS基础**

1. HTML属性操作

1、属性

属性由属性名和属性值两部分。属性的操作分为两种，一种是读操作，一种是写操作。

1. 读操作（获取找到相关信息）：元素.属性名；
2. 写操作（替换或者修改，但是不是添加）：元素.属性名=“要写入的内容”；

（1）innerHTML

读取当前元素内部的所有HTML内容，但是不包括标签本身。同样也可以修改或者替换元素内的内容。如果是在innerHTML后面没有其他内容，那么这个语句就是获取元素内部的内容；繁殖，如果是添加了内容跟在后面，那就是向当前元素中添加内容。

在添加内容的时候需要注意的是，如果元素内不能原来就有内容，那么添加之后就会将原来的内容覆盖掉。

X.innerHTML=“新内容”；

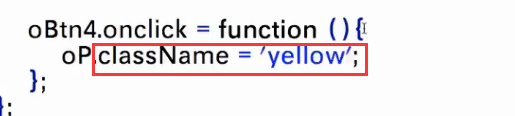
1. outerHTML

这个属性在获取元素内容的时候是把选中元素的包括标签在内的东西全部选中的。也就是说在获取内容的时候是讲结构一起取出来的，但是innerHTML就只是取出内容，不要结构。设置值的时候就是直接设置到显示的页面上。

1. innerText/outerText

这两个方法在获取内容的时候是一样的，都是获取元素内部的文本内容，不获取结构。在设置值的时候，后者是直接改变显示在浏览器中的内容，但是如果弹出来看的话，依然还是其原来的内容。

1. 属性操作注意事项
2. JS中不支持“-”，因此，但凡遇到这个符号连接的属性等，都要将其去掉，第二个单词首字母大写（使用驼峰命名法）。
3. Js中进行样式操作的时候，要是需要改变的样式比较多的时候，最好的办法是写成CSS的class，然后给一个事件，进行样式替换（替换className）。

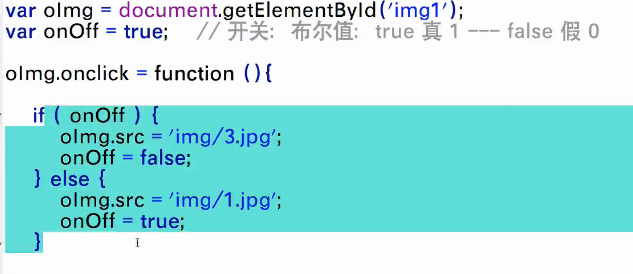


1. class在JS中是保留字，因此子命名的时候需要注意避开。保留字就是js留下来，以备后续发展会用到以成为关键字的保留部分。

**（4）关于判断**

1. 所有的相对路径都不能用来做判断。

但凡使用都会出错；但是绝对路径是可以的。有时候需要判断，但是又不能使用相对路径，那么最好的方式就是把判断条件转化一下，引入一个第三者进行迂回实现。例如：

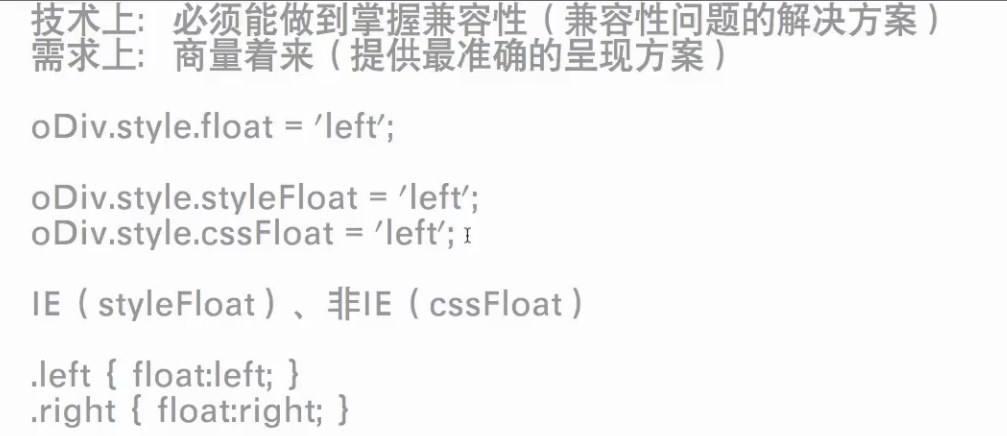


1. 浏览器某些默认样式，如颜色值也不能用来做判断。
2. innerHTML的值也不能用来做判断条件。
3. 背景色的值也不能用来做判断。
4. 关于表单元素

IE6、IE7、IE8不支持更改表单元素的类型，因此为了不出现兼容性问题，最好不要随意更改表单元素的类型。要是实在需要使用这样的不同的类型，可以选择显示、隐藏来实现。

**（6）关于样式的兼容**

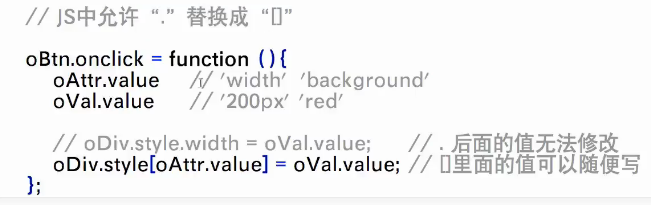
在IE和非IE的浏览器中，总是存在很多兼容性问题的，比如下面的例子。这样的问题，有时候可以通过写一些通用的东西，在需要的时候来进行改写，这样就可以做到兼容或者的到更好的结果。



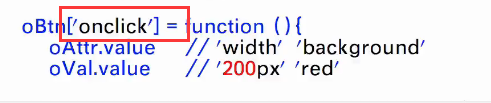
（7）中括号的使用

在JS中，有两种运算方法，一种是‘点号’，一种是‘方括号（中括号）’。本来在大多数时候是可以相互通用的，但是在区别就是前者（点号）后面的值是不可以通过js程序操作来进行更改，而后者（方括号）后面的值是可以通过js来进行更改的，因此也就把前者看作静态的，后者看作动态的。也就是说，使用方括号语法可以将属性值设置成变量，但是点号语法是没办法实现变量为属性赋值的。

如果对象的名称是那边来那个的时候必须要使用方括号来访问。



注意，方括号中的字符串是有引号的。



1. 条件判断
2. 在js中条件判断的方法主要有：if、if……else、if……else if、switch（）{case……}等四种方法。
3. 其中比较常用的if...else方法还可以使用三目运算符来等效替代。
4. 能够用来进行条件判断的值有：
5. 布尔值；
6. 非零数字，非NAN（0或NAN）；
7. 非空对象（null和undefined为空）；
8. 非空格字符串/空字符串

注意之前的那些不能用于条件判断的内容。

1. 数组应用以及实例

数组可以用来存放很多变量，但是最好每个数组的内部存放的数据类型是一致的，否则容易造成混淆。想要获取数组中的数据，那就用方括号语法，通过下标来访问。

1. 数组的创建

var arr=new array()//创建一个数组

var arr=[];

这是最常用的创建数组的方法，也可以在创建数组的时候将其长度定义好，但是在实际使用的时候是不会有多大影响的。因为如果在使用中的时候长度不够的话，数组会自动进行扩容，保证数据能够正常保存进去。



1. 数组元素的访问

在访问数组元素的时候可以通过下标来直接访问，也可以通过等号来为数组元素就会in行赋值。

1. 数组元素的添加与删除
2. 、添加
   1. push（value）这个方法是将内容添加到数组的最后面，并返回一个新的数组长度。
   2. unshift（value）这个方法就是向数组的开头插入内容，之前存在的内容自动往后挪一个位置，然后返回一个新的数组长度。
3. 删除
   1. pop（）方法删除最后一个元素病返回被删掉的那个元素。
   2. shift（）方法是删除数组最前面的那个元素，然后将删除的元素返回。

（3）splice（）

这个splice（）方法不仅可以进行内容添加，还可以进行删除。

1. 添加（插入）：添加的时候就是传入两个位置参数，然后就是需要添加的内容（num1,0，value）num1指的是要插入内容的位置，0表示删除0个，value表示要添加的内容，这个内容可以是一个值，也可以是多个值。
2. 删除/替换：（num2,count，value）num2指的是要删除内容的位置，count表示要删除元素的个数，然后如果有value值的话就是进行替换操作，否则就是进行删除。
3. 截取：slice（start，end）两个参数分别是截取数组元素的开始和结束的位置，返回一个新数组，如果end为空，那就是赋值start后的所有内容。
4. 合并：concat（arr1，arr2...）就是将多个数组连接起来，然后返回一个连接后的新数组。
5. 拷贝：
   1. slice（0）：返回一个新数组，不是指向；
   2. concat（0）：返回一个新数组；
6. 排序

sort（function）里面传入一个比较函数，决定排序的方法；其中的比较函数可以是根据需要编写。

reverse（）将数组反序。

1. 数组字符串化

join（）方法，参数就是连接数组元素为字符串的符号，如果是空引号的话就是直接无符号连接成字符串。

（9）字符串转化为数组：split（）

使用空引号对字符串进行分割，然后返回的是一个分割后的数组。

1. 数组的三个属性
2. Length

表示数组的长度也就是数组的元素个数，数组的索引是0开始的，所以一个数组的上下限分别是0和length-1.

1. prototype 属性；
2. constructor 属性；

第三章循环

JS中的循环语句主要是for、while和do…while

1. 获取元素的方法

getElementByTagName方法，这种方法获取到的元素实际上应该是一个数组，而不是像ID方法获取的一样是一个唯一的元素，这是由于ID具有唯一性，因此通过ID获得的与元素当然就是唯一的。

而在这个方法中，通过元素标签获得的元素是不具有唯一性的，因此以一个数组的形式来存放；因此，在访问的时候也是需要用访问数组的方法（方括号语法）来访问的。

从二者功能上讲，ID方法可以将其视为静态方法，因为他只能在document后面，但是上面的方法就可以在很多元素后面，比ID更加灵活。

1. for循环

解决的问题：解决了相同代码多次执行时候的操作简化。要是需要对二维数组进行访问，方法和C语言的是一样的，使用两层循环就可以实现 。

1. while循环

while（exp）{//要执行的代码}

exp为一条判断语句，最终结果可以使用一个布尔值来表示，true就继续执行，否则就停止循环，执行后面的代码。

1. do…while（condition）

执行的时候大体上和while的相同，唯一的不同就是这个循环方法至少会执行一次。

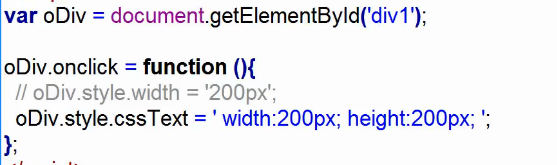
1. 结束循环的两种方法
2. Continue——结束本次循环，继续下一次循环；
3. break——结束当前的循环体，继续执行后面的内容。

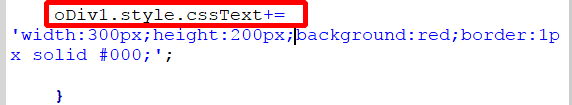
性能问题：

1. 循环数组的数组长度X.length,最好的办法是将其存进一个变量里面去，免得再循环中要多次计算长度，这样会影响性能；
2. 和上面的类似，对某些每次循环都需要的东西，并且每次使用都会计算的时候，最好的办法就是现将其计算出来存入变量中，然后再使用的地方直接使用哪个变量，这样会减少计算机的计算量；

6、cssText的使用（类似添加class）

这个方法是一组样式属性及其值的文本表示。这个文本格式为一个css样式表，去掉了包围属性和值的元素选择器的花括号。这个方法是给元素设置样式的，如果是使用“+=”他的作用是添加样式，如果是“=”那就是替换的效果了。

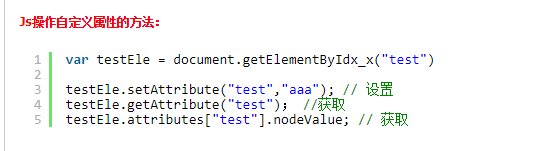




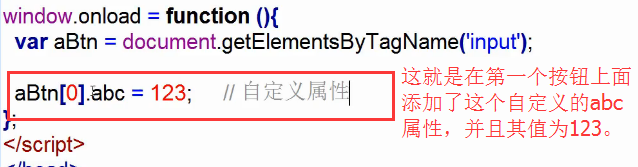
但是，这种方法了解一下就好了，实际上在使用的时候会降低性能，因为每当样式发生改变的时候都有可能引起重绘或者是重排，因此并不如使用“修改class”的方法实在。

1. 自定义属性

1、自定义属性的方法



在JS中设置和获取自定义属性的方法如上图所示，这是比较标准的写法，下图所示的是比较简单的书写方式：



同时，不仅可以添加自定义属性，而且还可以改写。要是对多个元素添加自定义属性的话，那么在循环中进行设置就可以了。

JS可以给任意的HTML元素添加任意多个自定义属性。

下面是一个通过添加自定义属性来实现变换li的背景图片的例子。为了实现多个li之间不会相互影响，在此处就需要使用判断并且还要给每一个li添加一个指只属于他自己的onoff属性，以此来判断当前li所处的状态。这种方法和运动中，多个div的运动是一样的。

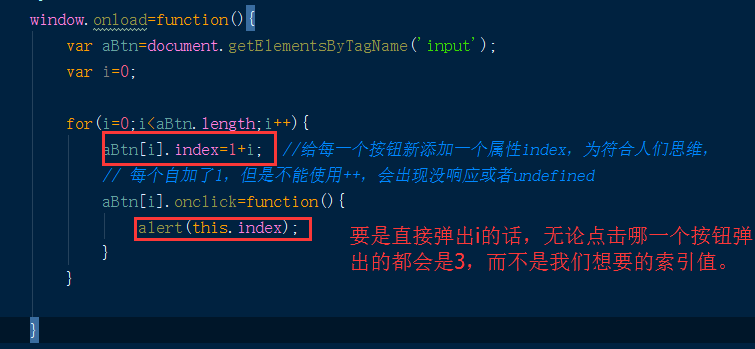
其中的onof就和开关的作用是一样的，进行切换以此执行不同的代码：



1. 添加索引值（index）、匹配数组、HTML元素

需要注意的是，在循环中包含有函数的时候，不能随便调用循环变量，比如‘i’等，因为就算调用，十有八九也得不到正确的值，以长度为3的数组为例。原因如下：

首先，for循环执行完成之后i的值会改变，利用alert弹出i的值会是3，因为在循环中，最后结束的时候i自加了1，因此就会是3。其次，因为i的作用于是整个for中，也就是说只要for中改变了i的值，最后弹出的就是i改变以后的值。但是这里我们利用index将i变为索引值将其添加到控件对象上，在我们需要的时候再去按照索引查找调用，也就是点击的时候直接触发。



1. JS的数据类型

一、数据类型

1、基本数据类型

JS中有六种数据类型，其中的五种是简单数据类型，分别是undefined、null、string、number、boolean，以及一种复杂是数据类型object，其实质就是一组无序的名值对组成。

2、变量

使用var来定义变量，并且由于js是弱类型语言，因此无法确定变量一定是什么类型，其类型是随着存储的内容变化的。

1. left声明变量

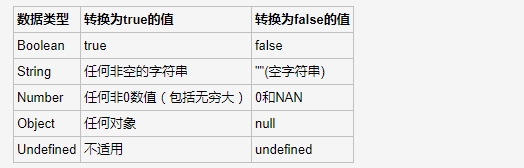
left方法声明的变量只能在left所在的代码块中有效。

1. const

const方法来声明的变量一旦声明结束，那么久不能再进行修改。

二、数据类型相关操作

1. 数据类型
2. typeof：用来判断数据类型！！！它的返回值就是六种基本的数据类型。注意，typeof null的返回值是object，因为null被认为是一个空对象引用。
3. 判断一个对象是不是数组的方法是：instance of ，如果使用typeof的话将会返回object而不会判断出是不是数组类型。
4. undefined类型的值只有一个，那就是undefined，在声明变量但是没有对其初始化的时候就会出现undefined的返回值。
5. null类型是第二个只有一个返回值得数据类型，那就是null。从逻辑上来看，null是一个空的只针对象。
6. boolean类型只有两个字面量，true和false。



使用一个感叹号是用来转换布尔值，使用两个感叹号（！！）则是将后面的表达式强制转换为布尔类型的数据，值就是和原值等价的布尔值。

1. number

这种类型表示整数以及浮点数。还有一种特殊的类型就是NAN（非数值），用来表示本来要返回数值的操作而未返回数值的情况。在js中，任何数除以0都会返回NAN。同时，NAN很特殊，它和任何数都不相等，包括NAN自身。

1. string

字符串可以用双引号或者单引号表示，用于表示零或者多个16位Unicode字符组成的字符序列。它的值是不可变的，可以将其看成是和基本引用类型相类似的不可变引用类型。

（8）.length：计算字符串、数组等的长度，字符串中，空格也会被统计在其中。

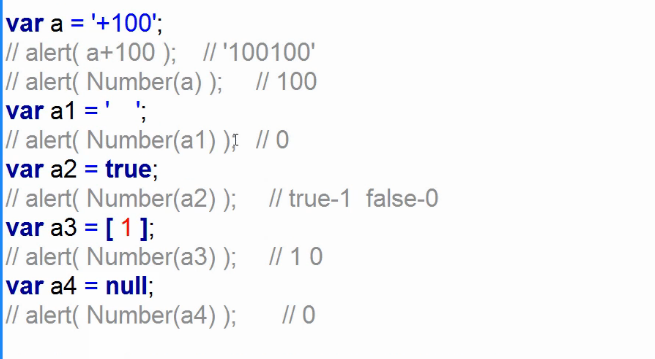
（9）function属于object的派生类型中的一种。

charAt（）：用来返回所查找的字符在字符串（数组）中的位置。

三、数据类型转换

1、数据类型转换——Number（）

这个是由ECMAScript提供的方法，是将字符串转化为数字，就是最大化地转换数据类型。在转换空字符串、空数组的时候会转换为0，要是数组中放了数字或者字符串的话，就按照起自己的规则来进行转换。



Json、函数、undefined等都无法进行类型转换，因此返回的值都是NAN，也就是非数字类型。Json和数组还有区别，即使json是空的也不会和数组一样返回0，而是NAN。

2、转换成布尔值

手动转换为布尔值的方法有两种，一是使用两个感叹号强制转换，另一种就是使用Boolean（obj）进行转换，但是前面不能加new，否则就变成了声明一个boolean类型的变量了。

3、显式类型转换

(1)Number（str）转换——将字符串str转换为数字类型；如果本来就是数值类型字符串，那么返回的就是本身；如果不是一个数值类型的字符串，那么在转换的时候就是NAN；如果要转换的内容是空字符串，那么转换的结果就是0.

(2)parseInt()——忽略前面的空格直到找到第一个非空字符，还会将数字后面的非数字字符串去掉。如果第一个非空字符不是数字的话，那就结束并返回nan；如果第一个是数字的话就找到往后存在的连续的数字，直到下一个不是数字为止，并且计算后面还有其他数字也不会继续进行转换了。这个方法得到的是整数类型的结果。

（3）parseFloat（）过程和上面一样，但是返回结果是浮点型数据。

（4）转字符串——将其他数据类型转换为字符串

String（obj）；

toString（）——undefined和null不嫩使用toString（）方法。

1. 函数

一、函数创建的三种方法

1、函数声明

function sum1（参数）{ //执行代码； }

2、函数表达式，又叫函数字面量

var sum2=function（）{ //执行代码； }

上面两种方法的区别是，函数解析器会先读取函数声明，并使其在执行任何代码之前可以访问；函数表达式则必须要在执行到函数字面量所在的代码才会真正被解析执行。

3、自执行函数严格来说也是函数表达式，主要用于创建一个新的作用域，再次作用域内声明的变量不会和其他作用域的变量冲突混淆，大多是以匿名函数形式存在的，并且会立即自动执行。

下面是一些自执行函数：





4、函数构造法，参数必须加引号

var sum3=new function（‘参数1’，‘参数2’…，‘执行的代码…’）；

按照我们希望高效的愿景，使用这种函数构造法是不科学的，效率很低。因为这种方法会将函数解析两次（县解析常规ecmascipt代码，另一次是解析传入构造函数的字符串），从而影响性能。

1. 函数传参、参数类型

参数指的就是JS 的数据类型，包括数字、字符串、对象、布尔值、函数、未定义；

传递有名字的函数的时候是不需要带括号的，但是匿名函数就需要了。

1、函数的参数和其他语言不一样，函数不会考虑参数传入的个数，也不在意函数是否有参数，以及参数是什么类型。

2、js函数中可以传入同名形参，但是最后起作用的一定是最后传入的一个，在严格模式下，出现同名形参会出现错误。

3、档实参比声明的参数要少的时候，那么剩下的形参都会设为undefined值；当实参个数比形参多的时候，那就会使用到arguments对象来保存参数。

4、函数的参数是以数组的形式保存的，函数接收到的始终是这个数组，而不关心参数的数量以及类型。如果需要访问这个数组就需要使用arguments对象，但是arguments对象并不是数组实例而是一个类数组，可以使用方括号语法想访问数组一样来访问这个类数组；也可以使用length属性来获得这个对象中你保存的形式参数的个数；形参只是提供便利，但是并不是必需的。

5、函数重载

Js函数并不能想传统意义上那样实现重载，js函数没有签名，因为参数时由包含0或者多个参数的数组来表示的，而没有签名，真正的重载就无法实现，只能通过检查传入函数中的参数类型和数量并作出不同的反应来模仿方法的重载

6、参数传递

Js中函数的所有函数的参数都是按值传递的，也就是说函数外部的值复制到函数内部的参数，就和把一个值从变量复制到另一个变量是一样的。

7、函数传参的应用

重用代码：

1. 尽量保证HTML代码结构一致，可以通过父级选取子元素；
2. 把核心主程序实现，用函数包起来；
3. 把每一组中的不同值找出来。
4. JS作用域

一、作用域和作用域链

1、作用域

作用域就是js中变量或者函数等的作用范围，这里的作用指的是进行读写操作。在JavaScript中的作用域有两种，分别是全局作用域和局部作用域。

2、全局作用域 ——在代码任何地方都能访问的到

（1）最外层函数和最外层函数外面定义的变量拥有全局作用域。

（2）所有未定义而直接赋值的变量自动声明为拥有全局作用域。

（3）所有window对象的属性拥有全局作用域。

3、局部作用域

局部作用域只在一个代码块中起着作用，最常见的如函数内部的变量，常称之为函数的作用域。也就是说，如果是在某一个小范围内定义的变量，那么他就只能在哪个定义的范围内部使用。

函数内部的同名变量将会取代全局变量。

4、作用域链

作用域链：由于JS的变量都是对象的属性，而该对象可能又是其它对象的属性，而所有的对象都是window对象的属性，所以这些对象的关系可以看作是一条链 。链头就是变量所处的对象，链尾是window对象

在JavaScript中一切都是对象。函数对象和其他对象一样，也有可以通过代码访问的属性和仅供js引擎访问的内部属性，其中一个内部属性就是【scope】，该属性包含了函数被创建的作用域中对象的集合，这个集合称之为函数的作用域链，他决定了那些数据能被函数访问。

二、

1. 作用过程

在js代码的执行过程中，我们可以虚拟地将执行过程分为最少两个步骤：

1. 预解析。根据关键字查找函数、变量等，然后将其放在一个虚拟的仓库中备用，在预解析的时候，若是遇到重名的变量和函数就只留下有值的那个函数，而不是变量，因为此时的变量是未定义的，和二者所处的位置前后无关；要是遇到了两个重名的函数，那就依据先后顺序，留下后面的那一个，前面的被覆盖掉。
2. 逐行读取代码；

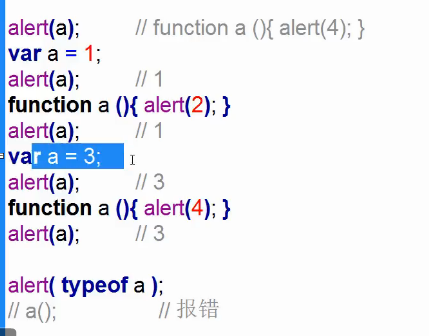
注意，在第一步中通过关键字查找相关信息的时候，对变量或者函数的内容是不会读取的。只有执行到了那一行，需要某一个变量或者函数的时候才会去读取其内部的内容，变量在正式运行之前都会被提前赋一个值undefined；函数在正式运行代码之前则是一个函数块，不会读取内容。

在逐行读取代码的时候才去读取变量或者函数的内容。

1. 表达式

但凡是那些能够改变变量的值的操作方式都可以称作表达式，例如一些运算符和Number（）等。表达式在逐行读取代码的时候是可以修改仓库中的值的，但是函数声明却不会改变这个变量的值。



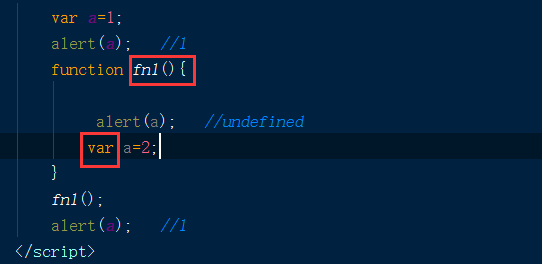


这是一个面试题目，问最后执行的结果。首先，根据ECMAscript的解析机制，会先去查找相关的变量和函数，然后存进虚拟的仓库中，实际并不存在。

开始逐行读取代码，因为在第一行查找的时候不会读取变量的值但是会读取函数块的内容。因此，在第一行代码alert （a）的时候，会弹出函数块的东西，然后在下一行遇到了表达式，由于表达式会改变变量的值，却不会改变函数的值，因此此时的a将会变为1，然后下一行的函数块不会被执行改变值的操作，再打印还是1；接下来变量值变为3，打印，然后进入函数块，a值不变仍然为3，接下来弹出a值为3，因此最后的结果就是3。

1. js是一个单线程的语言，因此需要按块逐步执行。例如，一个页面中含有两个script，那么它就会按顺序读取，先去读取第一个区域，接下来执行第一个区域的代码，然后再去读取第二个区域，执行第二个区域的代码。需要注意的是，在第一个script块中的变量等的值会保存下来（前提是没经过处理），在第二个script块中需要的时候也能够使用。
2. 局部变量和全局变量

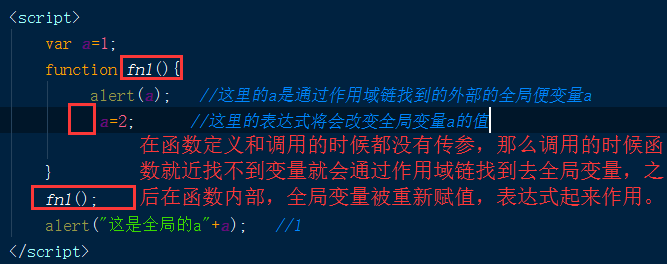
（1）



如上图，在一个script中，有全局的变量a=1，也有函数fn1。其中函数内部的定义的变量a就是一个只属于这个函数的局部变量。在执行这一段代码的时候，全局变量a的值是1，然后再执行fn1的时候，会进行这个函数内部的预解析，然后再逐行读取代码；所以在这个时候，函数弹出的a还没有被赋值，只是在预解析的时候被赋予的undefined，到了下一行才会被表达式赋值为2。

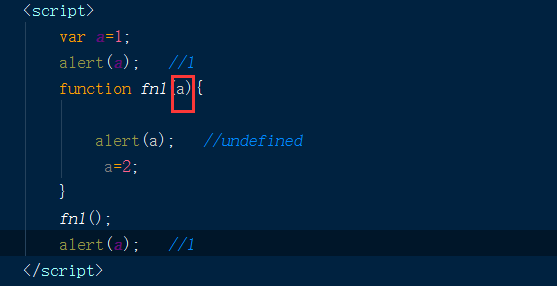
根据这个原理，就是局部的变量和局部的表达式只对局部进行操作，只在局部有效，不会超出它自己的定义范围，有作用域的限制。因此，在上面的例子中，函数中的a=2并不会影响到全局中的a=1的值发生变化。

（2）



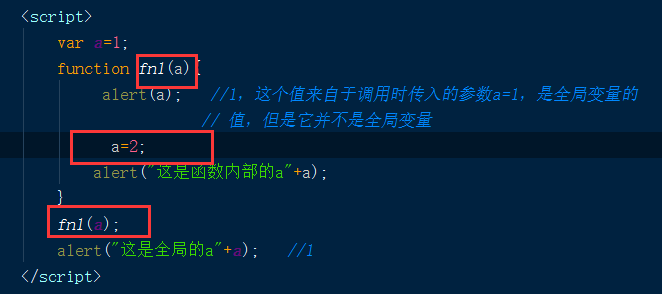
如上图，当局部中没有变量定义，但是存在全局变量的时候，那么就会出现局部的变量有能力更改全局变量的值的情况。首先定义了全局变量的值为1，然后 在函数中，在进行预解析的时候并没有找到局部的变量，因此函数会通过作用域链来到函数外部查找变量，找到全局变量a，这时函数调用的实际上就是全局变量a=1，然后再下一行中给a赋值为2，此时实际上就是在局部中更改了全局变变量的值。在接下来的代码执行中，弹出的a的值就是更改之后的值了。

（3）



如上图所示，在函数中传入的参数的本质就是一个局部变量，在调用函数进行预解析的时候，由于函数调用的时候并没有传入实参，那么在执行函数的代码块的时候就先给形参a赋值undefined，然后执行alert，最后再给a赋值为2，因为在接续函数块的时候，查找函数或者变量是依据“就近原则”的，因此在函数参数的地方找到了变量，那么也会就不会再去找到全局变量了，因此在函数执行的时候是不会改变全局变量的值的，所以在接下来的alert中，弹出来的值就是全局变量a中的值，而不是函数这个局部作用域中的a的值。

（4）



如上图所示，在调用函数的时候传入了参数，那么函数内部的局部变量在alert的时候就是外部传入的值。后面给a赋值为2也只是在函数内部改变了变量的a的值，并不会影响到全局变量a的值。

注意，在script中代码的读取顺序是从上到下的，但是在函数中是由里到外的，在函数中是不可以从外面去访问函数的内部（局部）变量的。

代码书写规范

函数的大括号才能是一个作用域，if、for、do...while等的大括号都不是作用域。

在火狐览器中不能对大括号括起来的函数进行预解析；但是除了火狐之外的浏览器都是可以正常使用的。但是，最好的做法是不要在for和if中去定义变量，不管是是局部变量还是全局变量。

三、闭包

JavaScript语言的特别之处就是：函数内部可以直接读取全局变量，但是在函数外部无法读取函数内部的局部变量。这就是JS所特有的链式作用域结构：子集会一层层向上找到所有父对象的变量，所以所有父对象的变量对于子对象都是课件的，但是反过来就无法成立。但是，如果在函数内部生声明变量的时候没有使用var的话，那么那个变量实际上是一个全局变量。

例如：这就是一个闭包的例子

function fn1（）{

var n=99；

function fn2（）{

alert（n）；//99

}

}

1、定义

闭包就是一个能够读取函数内部变量的函数；

由于在JS中，只有在函数内部的子函数才能读取局部变量，所以闭包又可以理解为“定义在一个函数内部的函数”

2、闭包的用途

用途有两个，一个是读取函数的内部变量；另一个是让这些变量的值始终保持在内存中不会在外层函数（fn1）运行之后被自动清除。原因就是fn2将值赋给了一个全局变量，而这个fn2依赖于fn1，也就是说，这个全局变量保证了fn1不会被垃圾回收机制清空。

3、使用闭包的注意事项

（1）由于闭包会让函数中的变量都保存在内存中，内存消耗会很大，所以不能滥用闭包，会降低性能。解决这个问题的方法是：在推出函数之前将不使用的局部变量全部删除。

（2）闭包会在函数外部改变父函数内部变量的值。因此，如果将父函数当做对象使用，把闭包当做它的公用方法，把内部变量看作是它的私有属性，这时一定要小心，不能随便改变复函数内部的变量的值。

817

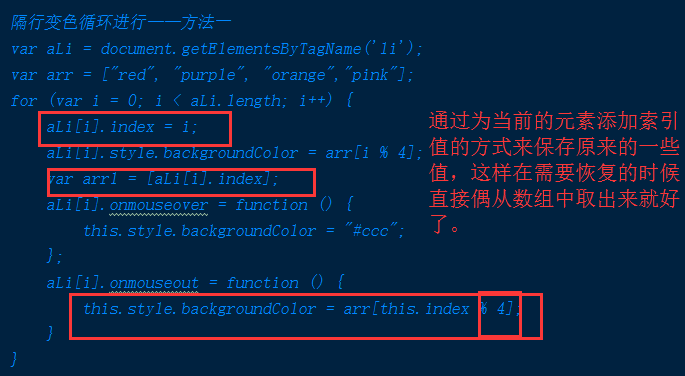
运算符 程序控制流

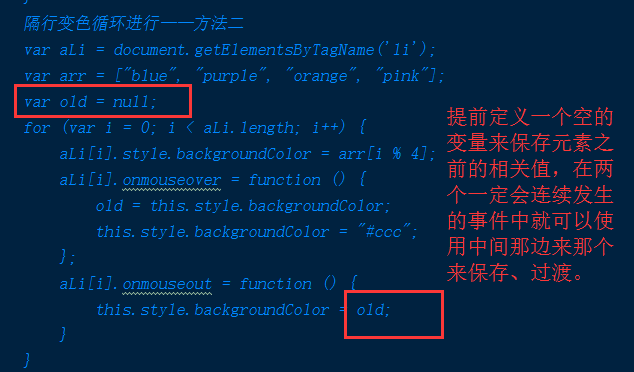
一、运算符

1、运算符+、-、\*、/、%

相关应用

隔行变色：通过取模来实现隔行的或者某个数为周期的变色。





反选或者全选



实现复选框的全选、全不选或者反选，就可以利用checked这个属性的值是布尔值来实现我们想要实现的效果。

全选就是让那些checked值不是true的复选框的该属性值全部变成true；全不选就是变为false；反选就是true和false之间切换。

二、程序流

1、定义

指的就是在程序代码执行过程中可以对程序进行控制。

2、程序流程控制

（1）switch

注意两点，一是每一个case后面再绝大多数时候都需要break，需不需要都要注意这个点；第二，最好有一个出错的default选项。

三目运算符

可以用来进行简单的判断，替换了if...else这样的格式，前提是代码数量不是特别多，要是太多了还是写if...else比较好。

循环for、while

这两个循环的执行顺序是一样的，因此在使用上都是可以相互换用的。

continue和break

break——跳出当前这个循环体；

continue——跳过当前这次循环；

程序中关于真假的问题——数据类型

数据类型：数字（NaN）、string、Boolean、object、function、undefined

真

非零数字（除了0和NaN）、非空字符串（含有空格也是true）、函数、能找到的元素、数组[]（即使是空数组）、json（{}）、；

假

空字符串、不能找到的元素、null、false、未定义、0、NaN

1. **定时器以及return**
2. 函数详解——return

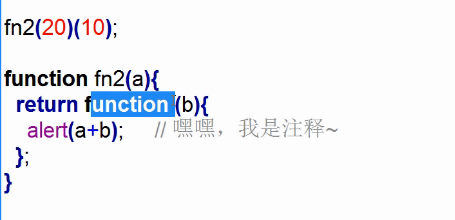
return返回的内容可以是数字、布尔值、函数、对象、元素、undefined等，后面跟的是什么，那么返回的就是什么。

注意，要是在return后面跟了一个函数名称（只有一个括号）的话，不会返回函数执行的结果，只会返回那个函数的代码本身。就连在该函数内部的注释也会一起返回。



因为函数名+括号即相当于return后面的内容，因此在上面的写法中，fn2（）相当于只是让其弹出函数名，但是并没有调用这个函数。因此，如果需要调用这个函数的话还需要在后面再加一个括号，写作fn2（）（）这种形式。

3、给匿名函数传参的情况



这里的两个括号相当于是在调用不同层级的函数，因此传进去的参数也会在其需要出现的位置被调用。

**函数名带括号和不带括号的区别**

函数名带括号，表示的是执行函数，如果是还有赋值的操作，意思就是将函数调用的结果赋值给这个变量；要是没有括号带有赋值操作，那就是将这个函数名称赋值给那个变量，但是并没有运行函数，也就不会有函数运行后的返回结果。

1. 所有的函数中都隐藏了一个return，在没有写明需要return什么内容的时候，所有的return都是undefined。一旦执行到return语句之后代码执行就会立即停止执行。
2. return：返回值
3. 函数名+括号=return（返回）后面（函数执行）的内容；
4. 所有函数默认返回值都是未定义；
5. Return语句后面的任何代码都不会执行了；
6. 函数详解——arguments
7. 实参

实参就是实际传递的参数；

1. 形参

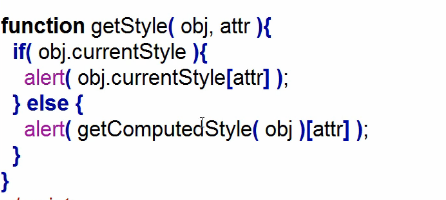
形式上的参数；

1. arguments——是一个类数组，而不是数组，是一个实参的集合。

它是一个类数组，因此是可以通过访问数组的方法去访问其内部的内容（实参）。

1. 使用情况：在函数的参数个数无法确定的时候，那么使用arguments就会很方便，使用方括号语法来获得相应位置的参数。同时需要注意的是，arguments中的变量值既可以读，也能够改写；而且它和变量的值是保持同步的，比如在arguments中更改了某个位置的值，那么与该位置相对用的变量值也会被改变。注意，这里的两个值并不是在同一个内存中，只是时刻保持同步而已。
2. getcurrentStyle和getComputedStyle获取样式

1、在js中，直接使用style来获取元素的时候是不能获取到的，它只能获取到行间样式。因此要是需要获取非行间样式的话，必须要使用另外的方法。



这里需要注意的是需要使用方括号语法来访问属性，只有这样才能更attr的值。并且获取到的是计算机计算之后的值。

还有，不要用上面的函数来获取复合样式，否则会出现兼容性问题，每个浏览器中这个函数的解析方法是有些不同的，因此需要注意。

不要获取没有设置的样式，这个也会存在不兼容的问题，因此需要避免这些问题。

以下是获取非行间样式的函数：

function getStyle（obj，attr）{

if（obj.currentStyle）{

return obj.currentStyle[attr];

}else{

return getComputedStyle(obj,false)[attr];

}

}

1. 定时器
2. 辨析

在循环和定时器的使用是由一些相似之处的，但是却有着就绝对的区别。循环时根据硬件性能的好坏，在一瞬间就完成，没有时间概念；但是定时器会有一个时间概念在其中，在一定的时间里循环做什么事，是根据事件来执行代码的。

2、函数需要调用的方法有直接调用、事件调用以及定时器调用。

1. setInterval（函数名（没括号），毫秒）；

注意后面的时间间隔（刷新频率）不要太小，最小的时间间隔不要小于14，这是jQuery写出的最小时间。

1. 清除定时器

clearInterval（定时器名称）；

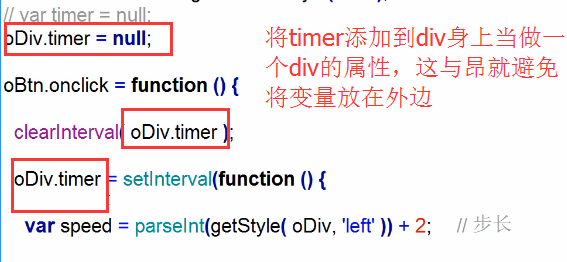
在开一个定时器的时候最好将定时器付给一个变量，这样在关掉定时器的时候更加方便。

1. setTimeOut（函数，毫秒）——这种定时器只执行一次
2. 清除方法：clearTimeOut（timer）；

注意：在使用定时器的时候，最好的办法就是将这个定时器变成对应元素的属性来处理，这样一是辨识度高，另一个就是使用的时候也比较方便，不会因为数目问题而出现冲突。（如下面的开关设置的内容）

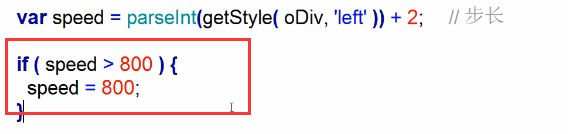
1. 定时器管理 函数封装
2. 定时器管理——开关设置
3. 变量外置

在大多数时候，将变量放在函数外边都不是最好的选择，这样会引出很多问题。因此，一个比较好的办法就是把那个变量当做都一个对象的属性添加到对象属性列表中，如下图所示。不再将timer当做变量来存放。



1. 关于边界的管理

如下，在赋值之前先进行判断，这样就避免在赋值的时候再进行判断，使元素二次渲染出现闪动。



1. 抖动原理以及实现过程

实现抖动效果实际上就是给那个元素的style们重新赋值，而且在进行重新赋值的时候需要注意，每一次完整的抖动是往一个方向动了多少，就必须往相反方向动多少。实现这样的循环赋值，那就用一个数组来存放这些值，这些值的变化从某种意义上来说也就是变化频率。

然后，既然涉及到了运动，那么就需要用到定时器，定时器的使用和之前运动的相关知识一致，不存在其他多余的东西。



其中的循环部分就是在给数组赋值，这里面的值就是在定时器执行代码的时候赋值给元素。在抖动的次数和数组的长度相等的时候，那么就关掉定时器，同时判断是否存在回调函数，要是存在回到函数就执行回调函数，要是没有回调函数就不结束代码执行。

1. **日期对象**
2. 系统时间对象

new Date（）得到的是当前代码执行的时候的系统时间。其中包含了所有时间相关的信息，包括年月日，时分秒和时区。

需要注意的是，月份是从0 开始的，也就是说每次使用的时候都要注意+1才能是我们需要的一个月份；同时，除了我们new的时间对象是object类型以外，其他的get方式获得的数据类型都是number。

其中从时间对象中获得日期或者时间的方式都是get加上FullYear、Month、Date、Day、Hours、Minutes以及Seconds（）；

使用new Dare（）的时候，可以在括号中添加日期来定义一个时间点。其中添加时间的方式有两种，一种是直接添加数字，另一种就是直接添加字符串。添加字符串的方法中，月份需要用英文单词，添加数字的方法中，月份则是从0开始计数的。

INew=new Date（2017,7,30,9,48,50）——对应的就是 年月日时分秒

INew=new Date（‘July 30，2017 9:50:10’）——对应的就是月日年时分秒



很多时候在使用事件的时候为了美观经常会将单个数字转换为两位，也就是在个位数字之前添加一个零。处理办法就是使用一个函数，在时间对象上获取到的时分秒之前以字符串的形式添加一个“0”，函数内部使用三目运算符比较简洁，当然使用if条件判断也是可以的。

**时间戳**

时间戳就是指从1970年1月1日0时0分0秒开始到现在的毫秒数。这个时间戳通常就是用作一种标记。它是时间对象的一种方法。

new Date.getTime（）；

**时间换算**

在使用时间的时候，通常需要进行换算，换算的方法就是：“先模上一级时间，再除以本级时间，最上层、最下层时间直接求模”以就是说，具体公式如下：

总时间t

天——Math.floor( t%86400 )；

小时——Math.floor( t%86400/3600 )；

分——Math.floor( t%86400%3600/60 )；

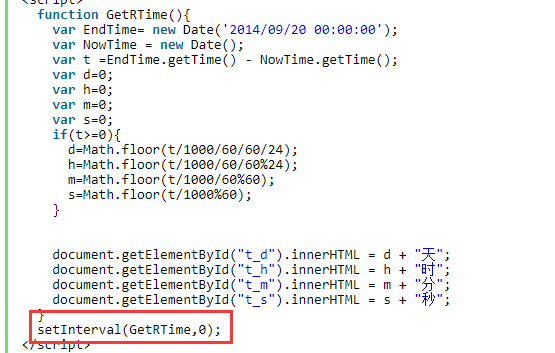
秒——Math.floor( t%60 )；

1. 倒计时

思路就是：写一个函数，里面定义两个对象，一个是倒计时的结束时间，一个是用来获取当前的时间，然后获得二者之间的差值，然后对这个结果进行换算出相应的时间来，比如天，小时等。

总之就是将来的时间不会变化，变化的是现在的时间。对于系统时间来说，都是以毫秒为单位的，也就是说在很多情况下使用的时候是需要将其进行转化的，这样对后面转化为小时和分钟都是比较方便的。

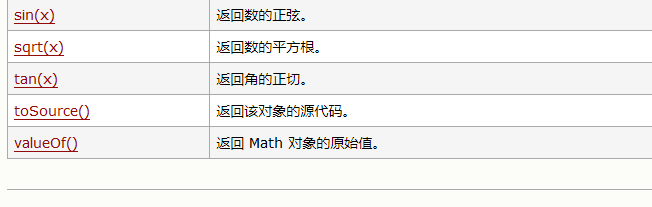
对于数字取整，方法有好几种，比如parseInt（）以及Math.floor()向下取整以及Math.ceil（）向上取整。



Math对象

Math对象并不是像Date和String这样的对象的类，它没有构造函数，不需要创建它而是直接像方法一样使用就可以了。





1. **字符串**
2. 字符串获取以及封装检测数字的方法
3. 字符串查询
4. 空格在字符串中也是算进字符串长度的。
5. charAt（）方法——动态方法

按照位置下标找到字符串中的字符。要是没有指定下标，那就默认找到第一个；要是超出了字符串长度的话，那就是返回null。

1. charCodeAt（）——动态方法

和charAt（）用法一样，只是它返回的是Unicode编码。

1. String.fromCharCode（）——静态方法

使用String对象来根据Unicode编码来返回一个汉字。要是添加了多个编码的话就可以添加多个编码，用逗号隔开。也可以使用循环来访问某一个范围之间的编码代表的内容。

1. indexOf（字符串，起始位置）

按照从左到右的方法挨个查找括号中要查找的内容，然后返回下标，就不用像charAt()一样还需要进行判断什么时候和需要查找的内容相等。

这方法接受两个参数，第一个就是要查找的内容，第二个就是开始查找的起始位置，如果该值为负数，那么就将其当成默认从零开始。（str.indexOf()）

查找开始后，如果没有找到指定的内容，那么就会返回-1，还有一种情况就是如果指定的查找的起始位置超出了字符串长度的话，那也会返回-1。正常情况下是返回的被查找字符串的开始的下标（从零开始的）。

千万要分清，indexOf（）方法返回的是一个下标，也就是说是一个数字而不是一个字符或者字符串。

1. lastIndexOf（）

这个方法的使用和indexOf（）是一样的，唯一的不同就是查找的顺序是不一样的，indexOf是从前往后（从左往右）找，而lastIndexOf（）是从后往前（从右往左）找。其他的都是一样的，包括参数的个数和意义。

1. 字符串比较

字符串比较实际上就是比较的是第一个字符的编码值的大小，当两个字符串中有一个地方比较出了大小，那么后面的就不会继续进行比较，直接结束了。

1. 字符串截取

**subString（）**

（1）str.subString（）——没有参数的时候就是默认为截取全部内容；

（2）str.subString（num）——从num位开始截取（获取）后面的字符串内容，第num位之前的内容都去掉；（**一个参数的情况**）

（3）str.subString（num1，num2）——从num1开始截取在num2之前的字符串内容

如果第一个数字比第二个要小的话，那么这个方法户自动将两个数字及交换位置，始终保持小的数字在前，大的数字在后，返回的是在这个范围内的内容。

1. 当有参数是负数的时候就会先将负数当做0来处理，然后对两个参数进行比较判断是否需要进行位置交换，如果需要就进行位置交换。

**Slice（）**

1. slice（num1，num2）方法在使用的时候，当参数出现“前大后小”的时候是不会交换参数的位置的，所以就有可能不能截取到内容；
2. 不会像subString（）一样将负数值的参数自动转化为0，而是将光标往后移（跳到后面），从后数几个，找到位置后从左往右截取字符串。
3. 在第一个参数是负数的时候，第二个参数不能出现正数。因为slice就算是负数也是从左向右开始就截取数据的，如果第二个参数是正数的话，就会出现截取不到内容而返回空内容的情况。
4. 字符串转换——只针对英文字符串

在很多情况下，获得的英文字符串用来进行判断的时候，为了避免出现一些未知的错误，通常比较保险的做法就是把它转换为大写或者小写，然后再用来进行判断。

1. 大写转换方法——toUpperCase（）
2. 小写转换方法——toLowerCase（）
3. 字符串分割——split（分割符，截取长度）

就是把一个字符串分割为一个数组。其中参数就是分割的标志。用作分割符的标志会消失，也就是说如果我们用字符串中的文字来作为分割符的话，那个字符就会消失。

分割符就是分割的界限，不会在数组中占位置，但是它的左右两边会占位置，即使左边或者右边并没有内容，也会在数组中占据一个位置，内容为空。

1. 如果在split（）方法中不添加参数的话，那么该方法就会将字符串转换为数组，整个存进数组中。但是，这种方法是不好用的，没什么特别的意义。

如果需要分割一个字符串但是又没有明确的子字符串可以帮助我们作为标志来进行分割，那么就可以直接在在参数位置传入一个引号（什么也不包含，包括空格），当做分割标志。这样就会把字符串分割为一个一个的字符存进数组。

1. 第二个参数是一个可选参数，如果需要限制自己分割得到的数组长度的话，那么就可以使用第二个参数来限定这个长度。
2. 字符串链接——join（）

使用join（）方法可以将一个数组中的把内容进行连接，如果不传入参数，那么连接出来的字符串就会默认加入逗号，如果传入空的引号的话就是直接连接而不加入任何符号；若是传入其他的符号的话就是以传入的符号为桥梁进行连接，将数组变成字符串。

综上所述，**split（）方法是将字符串分割为数组；而join（）方法则是将数组连接为字符串**。

**十三章**

1. Json（js对象标记，JavaScript object notation）
2. 定义

接送json就是一种数据格式，是用来存放数据的，它不包含任何控制结构，只用于数据传输。Json中数据类型包含字符、数字、布尔值（true/false）、null、对象，数组等。但是数组和对象是比较特殊并且常用的两种类型：

对象表示为键值对

数据由逗号分隔

花括号保存对象

方括号保存数组

Json语法是基于对象字面量和数组字面量的JavaScript语法。当使用字面量的时候将包含数组本身，但是不包括生成数组的表达式。

由于json自身轻便简洁的特点，所以在使用ajax的时候使用接送来传输数据是比较好的选择。

1. json格式

使用json键/值对方式来表示，键和值都用引号包起来便于数据在使用的时候进行转换，同时也保证数据转换不会出现错误；如果值是数字类型的话，那么数字类型不能加引号。否则会改变其数据类型，但是如果值是字符串类型就必须使用引号。

虽然也可以在键上不使用引号，但是这样不能保证数据转换的安全性，因此还是使用引号方式比较好。

Json的语法就是JavaScript的语法，所以在使用一些特殊字符的时候也需要进行转义：需要转义的符号有：引号（\’，\”）、空格（\b）、换行（\n）、换页（\f）、回车（\r）、制表符(\t)、反斜杠（\\）、正斜杠（\/）等

1. json的数据调用

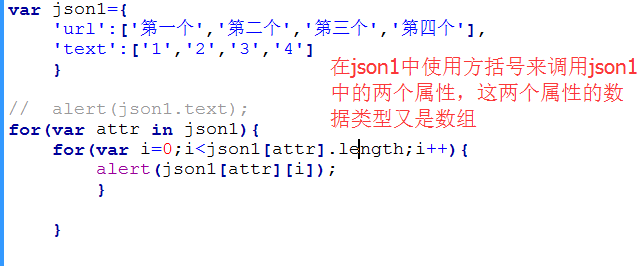
使用点号（.）来进行调用，然后可以进行调用、赋值等操作，但是不能使用方括号来进行点号替换。

1. json的特点就是可以存放很多类型的数据；也可以json放进数组里面，然后依次访问数组然后访问json就可以了。
2. Json读取——for...in 遍历

Json在使用的时候，经常会遇到那种json和数组嵌套的情况，一种是json包含数组，另一种是数组中包含json。

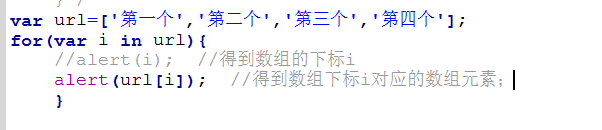
遇到这种情况的时候，那需要分清楚数据类型。如果是数组的话，那么就使用for来进行循环，访问所有内容；如果是json的话，那就使用for...in来访问json的内容。

这里有一个大括号和中括号的使用。使用大括号是访问的对象，而使用中括号是访问数组。但是，有一个情况就是，在访问json的时候也可以使用方括号，那就是使用属性来调用json的内容。



1. For...in

这个方法可以用来遍历对象身上的属性；也可以用来对数组进行遍历；



Json不能使用for循环遍历，但是数组是可以使用for..in来循环遍历的。

**json中的转换**

在数据流传输流程中，json是以文本，即以字符串的形式传输的；而js操作的是json对象，所以在使用的时候，json对象和json字符串之间的相互转换是关键。

Json字符串——var str1= ‘ { “ name ” : ” cxh ” , “sex “ : “male” } ’;

Json对象——var str2= { “ name ” : ” cxh ” , “sex “ : “male” } ;

1. json字符串转换为对象

var obj = eval（‘ （ ‘ +str 1+ ‘ ）’）;

或者

var obj = str.parseJSON()=var obj = JSON.parse(str); //从字符串转换为JSON对象

然后就可以直接用读取json的方式进行进行数据读取：

Alert(obj.name);

Alert(obj.sex);

注意：如果obj本来就是一个json对象，使用eval（）进行转换后，它还是json对象，也不会出现语法错误；但是，如果是paseJSON（）的话，那么就会抛出语法错误； 。

1. json对象转换为字符串

var last=obj.toJSONString(); //将JSON对象转化为JSON字符；

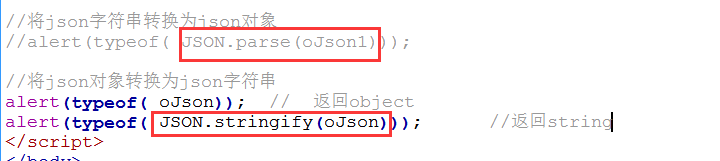
或者

var last=JSON.stringify(obj); //将JSON对象转化为JSON字符；

alert(last);

注意，高版本的json已经将json字符串转json对象方法重写为：string.parseJSON（）；将json对象转字符串重写为：toJSONString（）；

还可以使用下面的方法来转换：



1. 数组定义及清空数组效率问题
2. 数组的定义方法

Var arr=[1,2,3] / Var arr=[ ]

Var arr=new array(1,2,3);

第一种定义方法就是比较常用的定义方法，也是比较不容易混淆的方法，可以在定义的时候不传入数组元素的时候，那就是定义了一个数组长度不定的空数组。

在第二种定义数组的方法中，如果直接传入一个数字的话，那么就表示定义一个长度为那个数字的数组，若是同时传入多个数字就是表示定义了一个数组元素已经给定的数组；若是只传入一个字符（串），那也表示定义了一个长度为1的数组；

1. 数组的操作

如果需要比较高效地清空或者截断一个数组的话，那就是直接给数组的长度赋值，如果是要将数组清空，那么就将数组的长度赋值为零，那么就直接清空了数组的内容。

还有一种方法就是重新定义或者说是给数组arr重新赋值为[ ]空，这种方法相对给数组长度赋值为零效率要高一点，但是相对要简单一些。

1. 注意，在数组的一些操作中，数组的长度是既可以读又可以写的，但是同样有length属性的字符串就不能进行写操作，只能读。因此，在字符串中对其长度进行赋值是无效的操作。
2. 数组的部分方法
3. push（）

向数组的最后面添加内容（可以添加一个或者同时添加多个），返回的是数组的最新长度，而不是添加的内容；

1. Unshift

向数组的最前面添加内容，返回的值是数组最新的长度，不是添加的内容。在IE6、7中不支持unshift的返回值。

1. pop（）

删除数组的最后一个数组元素，然后返回删掉的内容；

1. shift（）

删除数组的第一个数组元素，然后返回的是数组被删掉的内容；

对于那些需要交换的值的特效，就可以使用这些方法来进行操作。在数组中取出来，然后塞到需要它去的位置，这样就可以实现数组元素的交换。

1. splice（num1，num2）——删除、替换、添加
2. 删除

arr.splice（num1，num2），其中num1是开始的位置，第二个数字是指定删除的个数；

1. 替换

arr.splice（num1，0，’\*’）其中第一个num1依然是指定要替换的数组元素，然后删除的数量是1个（就是num1那个元素），然后添加的的是第三个参数；（这个过程实际上就是一个先删除然后再添加的过程）

1. 添加

arr.splice（num1，0，’\*’，’\*’，’\*’），添加就是指定一个开始的位置，然后删除0个元素，接着在指定的位置后面添加任意多个新内容。

Splice的返回值只有删除的内容我们能够看到，实际上删除和替换都是有返回值的，因为替换也是有一个删除的操作在其中，添加的时候返回值是空的，因为实际上删除是空，因此返回值也就是空。

1. sort（）——数组排序

排序方式：冒泡、希尔、快速、归并、插入。。。

1. Sort（）排序默认是转换为字符串的方式按照Unicode编码值进行排序；
2. 比较函数

Sort（function（a，b）{

return a-b;

// return b-a;

}）

第一种情况就是，如果是a-b，那么就是按照从小到大排序；过程就是，如果a-b为负数，那么就不交换这个两个数的位置，如果a-b为正数，那么就交换这两个数的位置。相等的时候就不交换位置。

第二种情况就是，如果是b-a，那么就是按照从大到小排序；过程和上面相似，如果是b-a为正数，那么就交换两个数的顺序，如果为负数就不交换。相等的时候就不交换位置。

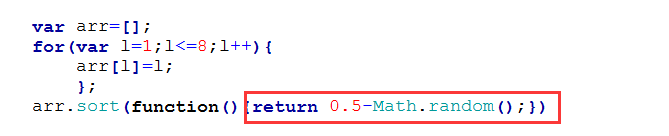
字符串转换为数字的方法就使用parseInt（）。注意，在sort（）排序的时候只是交换顺序，但是不会改变数据的类型。

如果要将本来很有规律的数组顺序打乱的话，就可以使用Math.random（）来产生随机数，然后再sort（）的比较函数中减去一个固定的小数，这样就会产生正负数不定的return结果，就会出现一会交换位置一会儿不交换的情况，这样就打乱了顺序。

**产生一组不重复的随机数的方法：**

主要的方法有三种，此处只记录一种，就是利用数组的sort（）方法来产生。

定义一个空数组，然后对其进行赋值（利用循环），然后再用sort（）方法来进行乱序操作，其中的比较函数是利用随机函数和一个小于一的数相减得到的，根据sort（）的性质可以知道，当返回的随机数正负不定的时候，数组中的数组元素是否交换就不会确定，于是便产生了一组随机数。



根据上面的做法，也就是说，如果要产生某一个范围类的随机数，并且数量有要求的话，使用数组是一种比较简单直接的办法。