Javascript整型数据定义：

可以是正整数、负整数和0

719 //十进制

071 //八进制的71（以0开头代表8进制）

0x987 //十六进制的987（以0x开头代表16进制）

\r代表回车符

Js中的string操作

Substr(start, [length] )

Substring( from, to)

检验一个字符串是否为空

String.replace(‘ ‘, ‘’).length == 0

Js调试，使用window.onerror,必须放在页面中所有js代码的最前面

<script>

Function myerror(\_message,\_url,\_line){

Alert(“错误信息:”+\_message

+”\n错误的URL：”+\_url

+”\n错误的行数：”+\_line

);

Return true;

}

Window.onerror = myerror;

指导在页面html加载结束后再调用这个js代码

Window.onload = function()

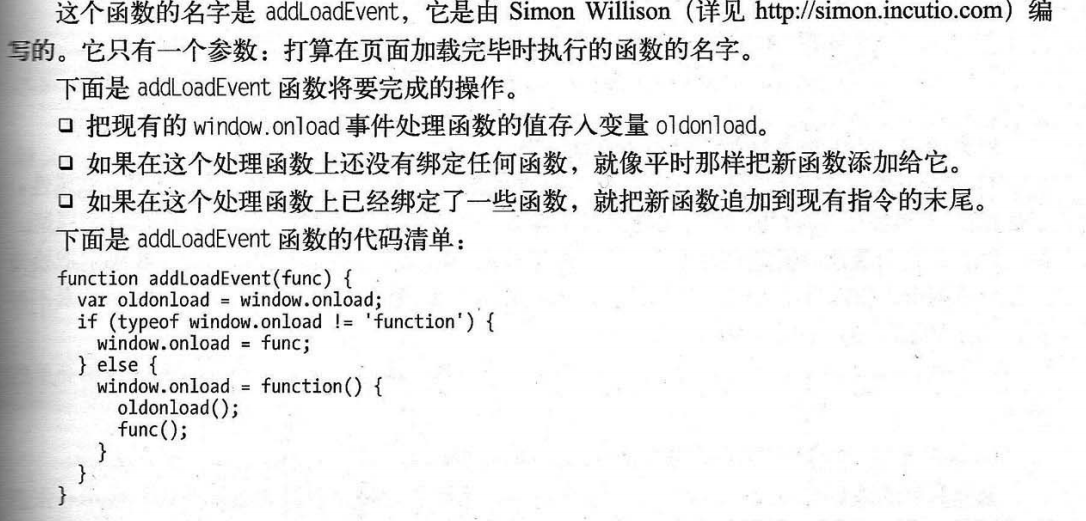
{ func1()

Func2()

…

}

Func1，func2的位置不作要求（可以在onload之前也可以在onload之后）



函数(特殊的对象类型)

1.命名方式一：function a(){…}

命名方式二：var a = function(){…}

(注意：此种方式中a代表的是这个函数，而不是函数的返回值)

2.参数:js中所有函数的参数都是可选的，如果调用时函数参数表为空，也是合法的但是返回值不一定合法。

3.若调用函数时输入的参数个数大于规定的参数个数，函数仍可以运行，忽略规定参数个数之后的所有参数

数据类型：

6个基本数据类型，一个对象类型

对象类型：name:value pairs（passing by reference）

基本数据类型：boolean

(passing Undefined（缺少值）:此处应该有一个值，但是还没有定义

by Null（没有对象）:一个值被定义了，定义为“空值”

value) Number：64位浮点

String

Symbol

===:严格相等（需要比较类型） “4”===4 false

==：值想等（不比较类型）”4”==4 true

对于布尔类型，

Boolean(false || null || undefined || “” || 0 || NaN) 返回值都是false

If(…||…||… … …||…)从左到右判断T/F，只要碰到第一个值为true的值，则停止判断，执行if之后的语句

Var x = “”||”hello”;

X = “hello” (默认选择第一个布尔值为true的变量的值)

对于对象类型：

为给定对象创建一个新实例： var Jeremy = new Person;

全局变量：可以在脚本中的任何位置被引用，（包括函数内部）

局部变量：只存在于声明它的那个函数的内部，作用域仅限于某个特定函数

在函数中声明局部变量若没有使用var，该变量会被认为是一个全局变量

Typeof操作符：检测给定变量的数据类型

“undefined”——未定义

“Boolean”——布尔值

“string”——字符串

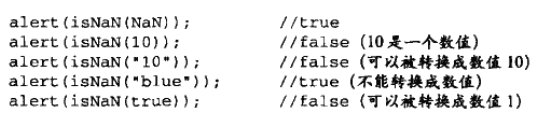
“number”——数值

“object”——对象 或者 null

“function”——函数

isFinite()函数， 判断参数是否位于最小与最大数值之间(-infinite ~ infinite)

isNaN() 判断参数是否“不是数值”



数制转换：

1. Number()用于任何数据类型

Boolean： true返回1，false返回0

数字： 返回原数字

Null： 返回0

Undefined：返回NAN

字符串：①字符串中只包含数字（包括前面带正负号或0），会转换成对应的十进制数字（前导0被忽略）

②浮点数转换同1

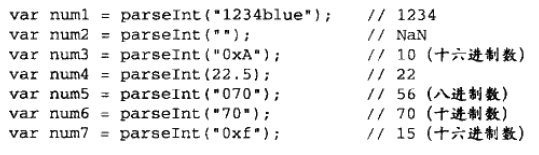
③字符串中包含有效的十六进制格式（"0xf”），转换为相同大小的十进制整数值

④字符串是空的，将其转换为0

⑤包含出上述格式之外的字符，将其转换为NaN

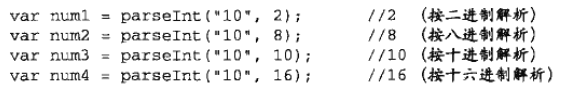
对象：调用对象的ValueOf（）方法，然后依照前面规则转换返回相应的值，若结果为NaN，则调用对象的toString方法，然后再次依照前面规则转换返回的字符串值

1. parseInt()函数：将字符串转换为整数，可以将不同进制的数字字符串转换为对应十进制



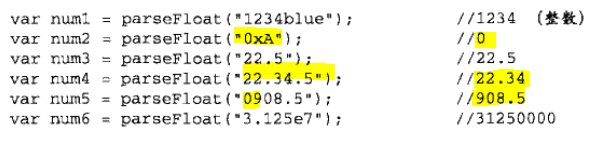
对于八进制数的转换存在分歧（0当作八进制前缀还是单纯的0），因此改进了parseInt（）方法

parseInt(“0xAF”, 16)第二个参数用于说明转换时使用的基数，此时字符串可以不带前面的0x



最好明确指定基数

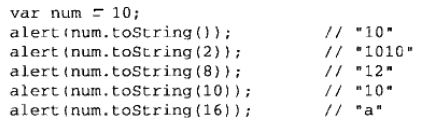
1. parseFloat()函数，字符串转换为浮点数



转换为字符串：

数值、布尔值、对象、字符串值 都有toString()方法，null和undefined没有

进阶：可以传递参数：输出数值的基数，可以输出任意有效进制格式表示的字符串值



String()函数

如果值有toString()方法，则调用该方法（没有参数）并返回相应结果

如果值是null，则返回”null”

如果值是undefined，则返回”undefined”

操作符：

1.位操作符：

有符号右移>>（左边补符号位）

无符号右移>>>（左边补0）正数是正常的，但是对于负数而言，这个操作会把负数的二进制码当成正数的二进制码，而且由于负数以其绝对值的二进制补码形式表示，会导致无符号右移后的结果非常大



AND 按位与

OR 按位或

Js中的构造函数:

Function Circle(radius) {

This.radius = radius; js中的构造函数没有返回值，否则返回一个空对象

}

Var myCircle = new Circle(10);

Console.log(myCircle);

输出为：Circle {radius:10}

某个类构造方法之外的方法

Circle.prototype.getArea =

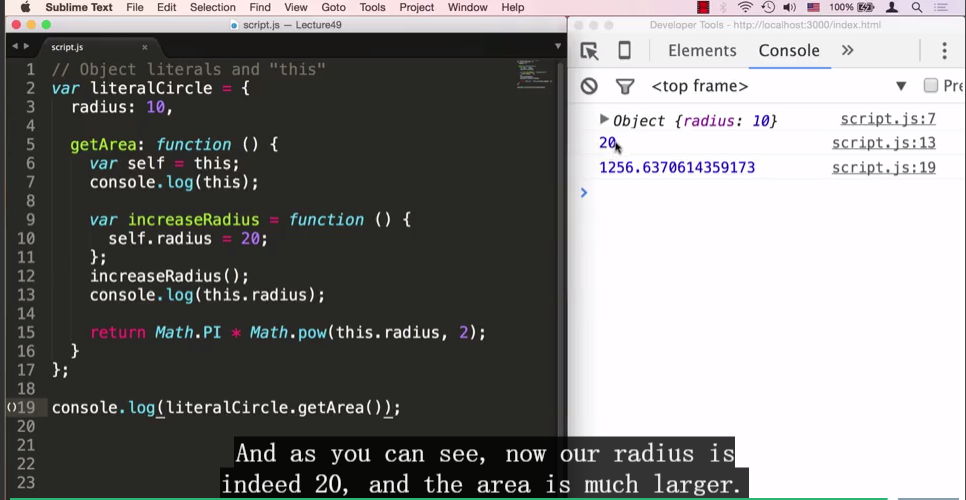
Function () {

Return Math.PI \* Math.pow(this.radius, 2);

};

Var Circle = new Circle(10);

Console.log(circle.getArea());



在函数increaseRadius中，其中的this指代的是window,而不是literalCircle

只有在increaseRadius方法之外先把this赋值给一个新的变量，之后其中才能获取到literalCircle的this值

Array

1.js中的array没有类型限制

var array = new Array();

var names = […, …, …, …];

可以是各种数据类型（对象+基本）+函数

2.array.length = array的长度

3.中间可以间隔不定义 如 array[0],[1],[2]都已定义，之后来个array[100]定义，也是合法的，未定义的部分看作undefined

4.遍历

①for( var I = 0; I < array.length; i++)

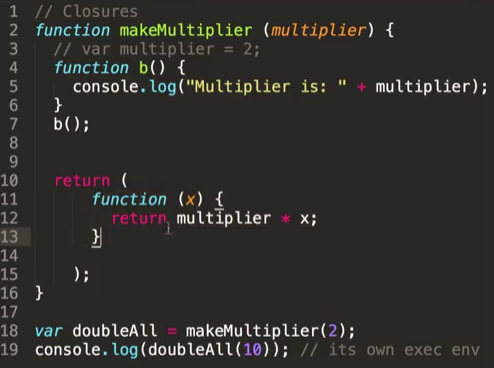
②for( var prop in array/Object)

For array: console.log ( array[prop]);prop看作下标

For object: console.log( Object[prop]);

Array是特殊的object

Closure



doubleAll代表的是返回的function(x)

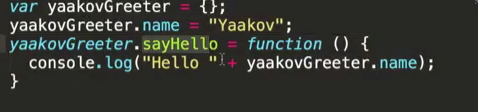
为什么这个function可以知道原先的multiplier？

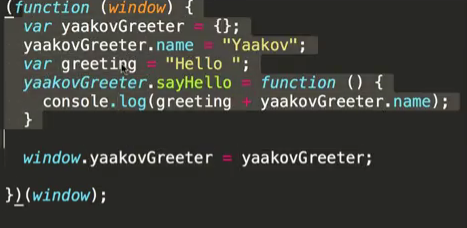
Js中内函数的作用域会保存其外部函数的变量

Js中的导入

在html中导入js文件，js文件中的全局变量导入到global namespace中，若几个js文件中的全局变量名重复，则以最后一个同名变量名的值为准

(若想避免这种情况，可以给每个js文件中的变量添加一个持有对象)





(IIFE Immediately Invoked Function Expressions)

带有src属性的<script>元素不应该在其<script>和</script>标签之间再包含额外的javascript代码，如果包含了嵌入的代码，则只会下载并执行外部脚本文件，嵌入代码被忽略。

文档DOM(document object model)

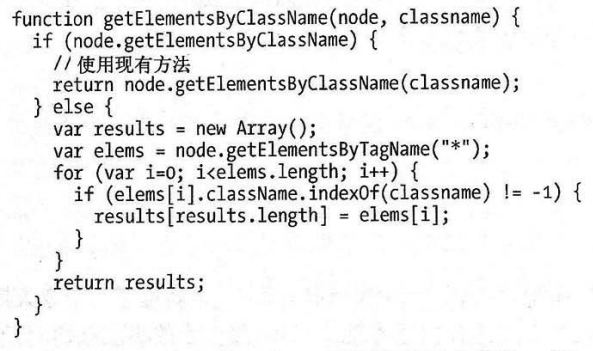
Document.getElementById(“…”).value获取对应id的值

Document.getElementById(“…”).innerHTML=… 向对应id处插入html文本

Document.getElementByTagName(“…”) 返回对应标签（\*代表所有标签））的所有元素，是一个元素数组

Document.getElementByClassName(class)返回对应类名的元素数组

自己实现的getElementByClassName ，node代表搜索开始节点，classname代表搜索的类名



Document.querySelector(“#title”) 以css选择器的样式选择文档内容，可选择id，class，标签等多种

获取和设置属性

Object.getAttribute(attribute)

Object.setAttribute(attribute, value)（修改不会体现在浏览器的查看源代码中，dom工作模式：先加载文档的静态内容，再动态刷新，动态刷新不影响文档的静态内容）

A标签，一般情况下点开都是弹出新窗口的，如果想要阻止弹出新窗口，在a标签属性的onclick中添加return false

标签的nodeType值，nodeType共有12种可取值，其中只有3种有实用价值

元素节点 nodeType = 1

属性节点 nodeType = 2

文本节点 nodeType = 3

可以让函数只对特定类型的节点进行处理

获取p元素的文本值，是<p>元素的第一个子节点

P节点.childNodes[0].nodeValue 等价于 p节点.firstchild.nodeValue

nodeValue可以用来设置节点的值

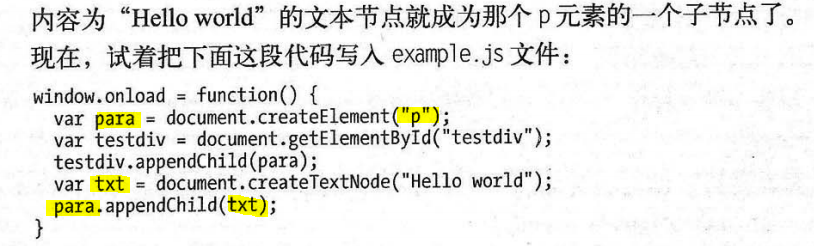
如何在js中新建html节点并插入

Var node = document.createElement(“tagName”);

nodeName.appendChild(node); 将node节点插入到nodeName下

其他新建节点的方法：

Document.createTextNode(text) 创建一个内容为text的文本节点；之后需要将这个节点appendChild到一个文本中，如p



将一个节点插入到某个节点元素之前insertBefore

parentElement. insertBefore( newElement, targetElement)

parentElement: 目标元素的父元素

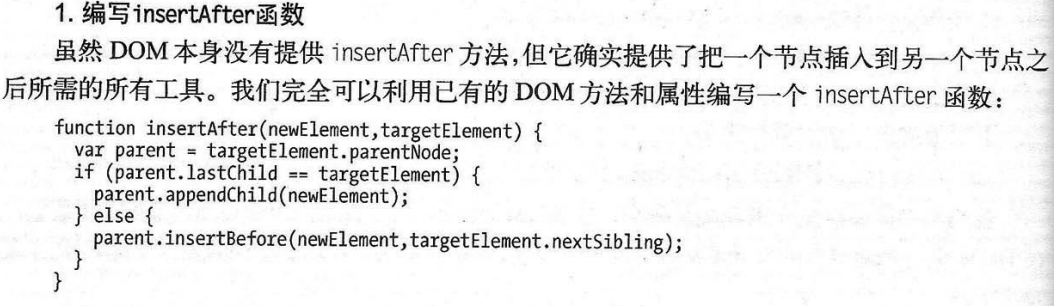
newElement：想要插入的元素

targetElement: 想把这个新元素插入到哪个元素之前

eg： var gallery = document.getElementById(“imagegallery”);

gallery.parentNode.insertBefore(placeholder, gallery);

将一个节点插入到某个节点元素之后



js创建新的浏览器窗口：

window.open( url, name, features)

url新窗口里打开的网页的url地址

name ：新窗口的名字

features： 以逗号分隔，内容为新窗口的各个属性

js平稳退化：应对某些禁用javascript的用户（如搜索机器人）

<a href=<http://www.example.com/> onclick=”popup(this.href);return false;”>Example</a>

Popup函数是根据输入的地址显示对应网页的函数，如果禁用了js，这个函数不能被使用，但是href所指向的地址也可以替代这个函数的作用

另外一种应对不支持javascript浏览器的方法：<noscript>标签，当js被禁用时，浏览器显示noscript中的内容，除此之外浏览器不会呈现noscript的内容

渐进增强：从最核心的部分（内容）开始，应该根据内容使用标记实现良好的结构，然后再逐步加强这些内容

对象检测：

If(method){ 相当于 if(!method) return false;

Statements

}

可以用来检测某个浏览器是否支持这个方法（注意：method后面不带括号，否则返回的是函数结果）

DOM Core：简化的html-dom语法

Document.getElementsByTagName(“…”) = document.tagName;

Elements.getAttribute(“src”) = element.src;

Elements.getAttribute(“href”) = element.href;

CSS-DOM

Element.style.color 可以获取对应元素的css中颜色样式

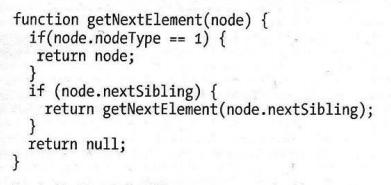
【注】不可以使用element.style.font-family 因为 js会把-解释为减号，加号、减号之类的操作符是保留字符，不允许用在函数或变量的名字里，改用驼峰命名

应该改为element.style.fontFamily

DOM style属性不能用来检索在外部CSS文件里声明的样式，也不包含在head部分style标签里声明的样式，style对象只包含在html代码里用style属性声明的样式

Element.nextSibling查找文章中的下一个节点（不是下一个元素节点）

如何查找下一个元素节点？



Js可以处理某些css难以处理的样式，如需要重复多次的任务（如给表格设置不同的颜色，对于那些支持css3的浏览器，可以使用tr:nth-child(odd)，对于那些不支持css3的浏览器，则使用js来对不同行做不同的处理）

可以实现css伪类的功能

通常不直接使用js修改style样式，可以通过js修改对应元素的类，再通过css修改对应类的样式，这样确保了网页的表示层和行为层的分离

Js动画

setTimeout( “function”, interval) 让某个函数在经过一段预定的时间之后再开始执行

function：函数名字

interval：经过多久后再执行第一个参数所给出的函数（单位：毫秒，1秒=1000毫秒）

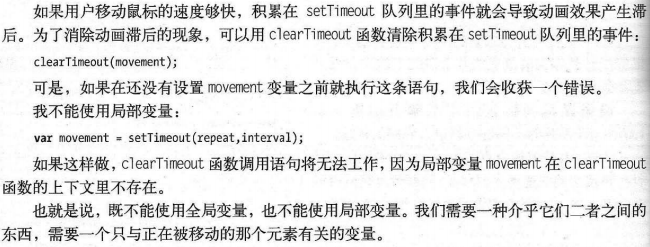
【注】最好把函数调用赋值给一个变量，这样之后可以使用clearTimeout函数来取消“等待执行”队列里的该函数，clear函数的上下文只能看到全局变量

Variable = setTimeout( “function()”, interval); （variable未使用var声明，是一个全局变量）

…

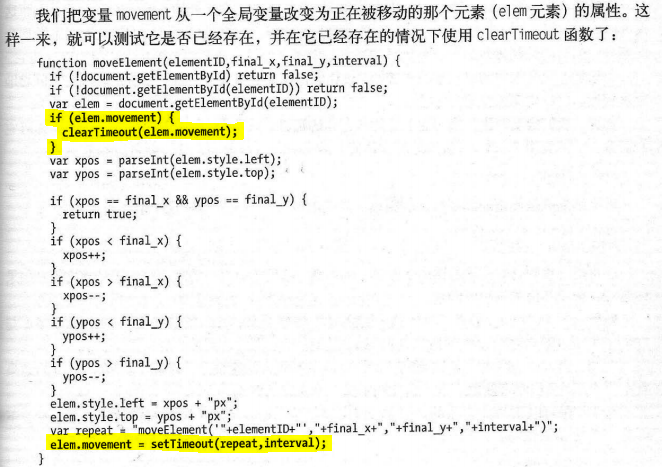
clearTimeout(variable);

【注】settimeout函数会维护一个等待队列，每次新加入的任务优先级都是最高的，不管上一次是否已经执行完毕，这个函数会被直接再次调用并试图执行新加入的指令，于是，当用户指令过快时，调用set的函数可能会试图同时执行两个相反的命令，使得动画出现问题，解决方法如下：



介于全局变量和局部变量之间，只与特定元素有关的变量：属性

Element.property = value（可以为元素创建属性）🡪专属于某个元素的变量



Math函数

Math.floor(number) 把任意浮点数向“小于”方向舍入

Math.ceil(number) 把任意浮点数想“大于”方向舍入为整数

Math.round(number) 把任意浮点数舍入为与之最接近的整数

Canvas画布

Ctx = Canvas. getContext(‘contextID’)返回一个用于在画布上绘图的环境，可以看做是画布 contextID当前唯一个合法值为2d

ctx.beginPath() 开始一条路径 ctx. closePath() 以绘制线条返回开始点

ctx.moveTo(x, y) 把路径移动到画布中的指定点，不创建线条

ctx. lineTo(x, y) 添加一个新点，然后再画布中创建从开始点到该点的线条

ctx. bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)绘制一条三次贝塞尔曲线



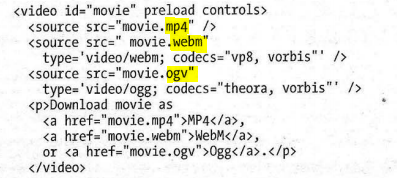
ctx. Fill() 填充当前的图像，默认颜色为黑色（ctx.fillStyle=颜色）

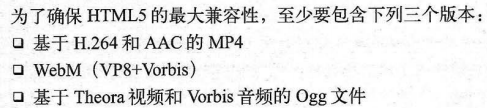
ctx. Stroke() 绘制已经定义的路径

ctx.strokesStyle 设置或返回用于笔触的颜色(color)、渐变(gradient)或模式(pattern)



Html5中的音视频处理：兼容性问题





使用DOM属性定义音视频控件的外观，例子见dom编程艺术page231

addEventListener(event, function, useCapture)



Js中渠道当前页面的url window.loacation.href

事件响应：

Document.addEventListener(“DOMContentLoaded”, function(event){

…

})

这段js代码表明js将在网页加载完成（在外部链接和图片加载完成之前）执行

GET

POST

JS中的response状态码

HTTP/1.1 200 OK : HTTP-version Response-status-code English-explanation

200 OK : 获取对方要求，并且拿到了对方所要求的数据

404 Not Found: 服务器找不到对方所要求的数据

403 Forbidden: 未认证客户尝试连接安全连接

500 Internal Server Error: 服务器发生未知错误

Ajax

Asynchronous(异步的) Javascript And XML

Ajax是异步执行的，即sendGetRequest的方法在执行中也会执行同级的其它代码，不能保证在ajax代码之后的代码就是在其后执行的，此时要把所有需要保证序列行的代码都加入到ajax中

Ajax应用主要依赖于服务器端处理，而非客户端处理

Hijax 渐进增强地使用Ajax

执行主体为XMLHttpRequest ,常用request.open(“GET”, “file name”, true);

参数： 第一个：GET/ POST/ SEND 指定请求类型

第二个：在服务器上将要访问的文件

第三个：指定请求是否以异步方式发送和处理

Request.setRequestHeader(“Content-type”, “application/x-www-form-urlencoded”);

在请求中添加application/x-www-form-urlencoded头部

这个头部信息对于POST请求是必须的，它表示请求中包含URL编码的表单。

Request.onreadystatechange是一个事件处理函数，它会在服务器给XMLHttpRequest对象送回响应的时候被触发执行

服务器在向XMLHttpRequest对象发回响应时，该对象有许多属性可用，浏览器会在不同阶段更新readyState属性的值，它有5个可能值：

0 未初始化

1 正在加载

2 加载完毕

3 正在交互

4 完成

Onreadystatechange事件处理函数在等到readystate值变为4之后，就可以访问服务器发送回来的数据

访问服务器发送回来的数据要通过两个属性完成：

①responseText，用于保存文本字符串形式的数据

②responseXML，用于保存Content-Type头部中指定为“text/xm”的数据

【注意】使用XMLHttpRequest对象发送的请求只能访问与其所在的HTML处于同一个域中的数据，不能向其他域发送请求

正则检测

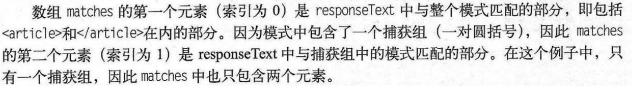
Text.match(正则表达式)返回包含各种匹配结果的数组

eg: 

\s代表任意空白字符

\S代表任意非空白字符

圆括号：捕获组



Matches[0]代表与整个模式匹配的部分，之后顺序代表每个 捕获组 匹配的部分

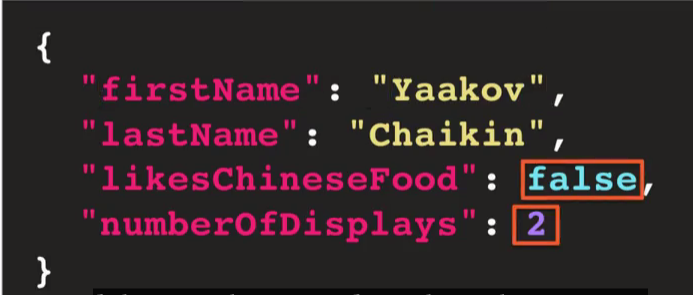
JSON

JavaScript Object Notation

轻量级数据交换格式

JSON的语法与js中的对象类型很类似，但是它实际上还是一个字符串

语法：属性名和字符串的值必须用双引号引起来



Converts from json string to object

Var obj = JSON.parse(jsonString);

Converts from object to json string

Var str = JSON.stringify(obj);

Js压缩代码工具（什么什么.min.js）

http://closure-compiler.appspot.com/home