

-

整理自互联网，仅供学习交流。



2018-10-30

[公司名称]

[公司地址]

目录

[一、 JS简介 1](#_Toc7444559)

[1.1 基本特点: 1](#_Toc7444560)

[1.2 日常用途: 1](#_Toc7444561)

[1.3 作用： 2](#_Toc7444562)

[1.4 js代码存在的位置 2](#_Toc7444563)

[1.5 基础语法 2](#_Toc7444564)

[1.5.1 定义变量: 3](#_Toc7444565)

[1.5.2 变量的打印: 4](#_Toc7444566)

[1.6 JS的基本类型和常量: 4](#_Toc7444567)

[1.7 解决表单重复提交 5](#_Toc7444568)

[1.8 JS中的运算符和Java中大多数相同: 6](#_Toc7444569)

[1.9 函数(function) 7](#_Toc7444570)

[1.9.1 函数的作用 7](#_Toc7444571)

[1.9.2 函数的定义和调用: 7](#_Toc7444572)

[1.9.3 函数的返回: 7](#_Toc7444573)

[1.9.4 匿名函数(经常使用): 7](#_Toc7444574)

[1.10 全局变量和局部变量 8](#_Toc7444575)

[1.11 系统函数 9](#_Toc7444576)

[1.12 面向对象 11](#_Toc7444577)

[1.13 函数参数的值传递方式 12](#_Toc7444578)

[1.14 内置对象 13](#_Toc7444579)

[1.15 数组 15](#_Toc7444580)

[1.15.1 为什么使用数组:存储多个数据. 15](#_Toc7444581)

[1.15.2 如何创建数组对象: 16](#_Toc7444582)

[1.16 prototype 17](#_Toc7444583)

123

# JS简介

JavaScript简介:

JavaScript一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为JavaScript引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在HTML（标准通用标记语言下的一个应用）网页上使用，用来给HTML网页增加动态功能。Node.js。

在1995年时，由Netscape公司的Brendan Eich，在网景导航者浏览器上首次设计实现而成。因为Netscape与Sun合作，Netscape管理层希望它外观看起来像Java，因此取名为JavaScript。但实际上它的语法风格与Self及Scheme较为接近。

为了取得技术优势，微软推出了JScript，CEnvi推出ScriptEase，与JavaScript同样可在浏览器上运行。为了统一规格，因为JavaScript兼容于ECMA标准，因此也称为ECMAScript。

组成部分:

JavaScript 的内容，包含以下三部分：

ECMAScript（核心）：JavaScript 语言基础(规定了 JavaScript 脚本的核心语法，如 数据类型、关键字、保留字、运算符、对象和语句等，它不属于任何浏览器。)；

DOM（文档对象模型）：规定了访问HTML和XML的接口(提供了访问 HTML 文档（如body、form、div、textarea等）的途径以及操作方法)；

BOM（浏览器对象模型）：提供了独立于内容在浏览器窗口之间进行交互的对象和方法(提供了访问某些功能（如浏览器窗口大小、版本信息、浏览历史记录等）的途径以及操作方法)。

## 基本特点:

是一种解释性脚本语言（代码不进行预编译）。

主要用来向HTML（标准通用标记语言下的一个应用）页面添加交互行为。

可以直接嵌入HTML页面，但写成单独的js文件有利于结构和行为的分离。

## 日常用途:

嵌入动态文本于HTML页面。

对浏览器事件做出响应。

读写HTML元素。

在数据被提交到服务器之前验证数据。

检测访客的浏览器信息。

控制cookies，包括创建和修改等。

## 作用：

交互式操作

表单验证

网页特效

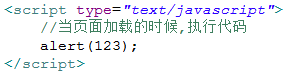
web游戏

服务器脚本开发

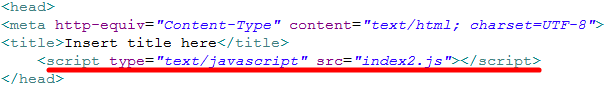
## js代码存在的位置

**JavaScript存在的位置**:

**方式1:** 在<html>标签中,任何地方添加<script></script>标签.标签中内容就是js代码. 虽然可以放在页面的任何地方,但是规范放在<head>标签中.

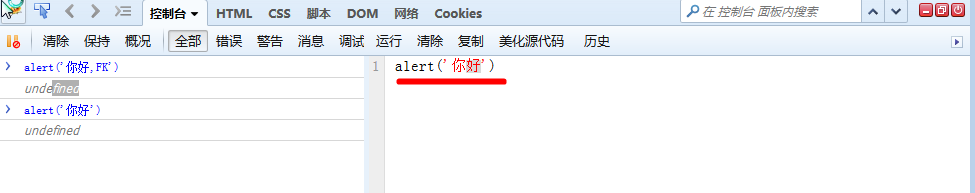


**方式2**:单独使用一个文件来编写javascript代码(**js文件**),在需要使用的页面中引入该文件.



**方式3**:把代码编写的**a标签**的href属性中,点击a标签的时候,就执行里面代码.





## 基础语法

基本语法(大多数和Java一样):

JavaScript中的标识符:

变量，常量，函数，语句块也有名字，我们统统称之为标识符。标识符可以由任意顺序的大小写字母、数字、下划线(\_)和美元符号($)组成，标识符不能以数字开头，不能是JavaScript中的保留字或关键字(和Java一样)。

合法的标识符举例：indentifier、username、user\_name、\_userName、$username

非法的标识符举例：int、98.3、Hello World

JavaScript严格区分大小写.

username和userName是两个完全不同的符号

JavaScript程序代码的格式

每条功能执行语句的最后必须用分号(;)结束，每个词之间用空格、制表符、换行符或大括号、小括号这样的分隔符隔开 。

语句块使用{}来表示

JavaScript程序的注释

/\*…\*/中可以嵌套“//”注释，但不能嵌套“ /\*…\*/”。 、/\*\*..文档注释.\*/

### 定义变量:

什么是变量，系统为之设置一个标识，程序可以用变量名来指向具体的对象内存，并通过变量名来获得对应的对象值。

**声明变量要使用var关键字**

例如：**var name;**//一般不使用name作为变量名

声明变量的同时为其赋值

例如：var name = "will";

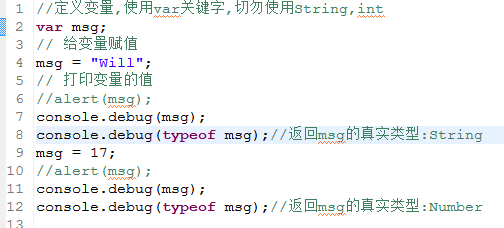
对已赋值的变量赋予一个其他类型的数据

例如：name = 17;

不事先声明变量而直接使用,报错 not defined.(未定义)

例如：x = 1234;

　提示：javascript定义变量无需指定类型，任何类型都使用var声明,感觉var就有点类型于java中的Object类型了

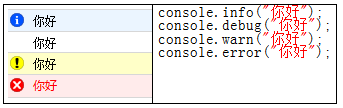


### 变量的打印:

**方式1:使用alert(内容)**;

**方式2:使用console.debug(内容)**; 注意:只有W3C的浏览器才支持(IE不行),Firebug中,Google的控制台.

好比System.out.println(内容);



## JS的基本类型和常量:

Number(数字类型)

1、整型常量(10进制\8进制\16进制)

十六进制以0x或0X开头，例如：0x8a。

八进制必须以0开头，例如：0123。

十进制的第一位不能是0（数字0除外），例如：123。

２、实型常量

12.32、193.98、 5E7、4e5等。

特殊数值：NaN(Not a Number)、Infinity(除数为零)，所对应的判断函数isNaN()、isFinite()

**Boolean(布尔值) true和false**。

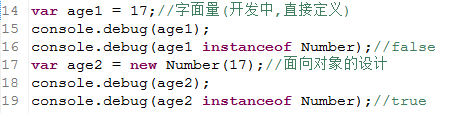
String字符串(注意js中没有char类型，所有'a'也是一个字符串)

“a book of JavaScript”、‘a’、 “a”、“”。

字符串中的特殊字符，需要以反斜杠（\）后跟一个普通字符来表示。

**null常量:表示引用任何的内存空间.**

**undefined常量 (未定义,定义未赋值)**



## 解决表单重复提交

|  |
| --- |
| <%@ page language="java" contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <title>Form表单</title>  <script type="text/javascript">  var isFlag = false; //表单是否已经提交标识，默认为false  function submitFlag() {  if (isFlag == false) {  isFlag = true;  return true;  } else {  return false;  }  }  </script>  </head>  <body>  <form action="${pageContext.request.contextPath}/DoFormServlet"  method="post" onsubmit="return submitFlag()">  用户名：<input type="text" name="userName"> <input type="submit"  value="提交" id="submit">  </form>  </body>  </html> |

除了用这种方式之外，经常见的另一种方式就是表单提交之后，将提交按钮设置为不可用，让用户没有机会点击第二次提交按钮，代码如下：

|  |
| --- |
| function dosubmit(){  //获取表单提交按钮  var btnSubmit = document.getElementById("submit");  //将表单提交按钮设置为不可用，这样就可以避免用户再次点击提交按钮  btnSubmit.disabled= "disabled";  //返回true让表单可以正常提交  return true;  } |

## JS中的运算符和Java中大多数相同:

算术运算符

赋值运算符

比较运算符

逻辑运算符

三元运算符

位运算符

------------------------------------------------

**JS中特殊存在的案例**:

**第一个不同**:

在JS中,比较相等的运算符: **==,===**.

问题:=和==以及===的区别:

= : 和Java相同,用于给变量赋值.

== : 用来比较变量或者常量的内容,不管数据类型.

===: 比较数据类型和内容,若数据类型相同,且内容相同返回true.

console.debug(17 == "17");//true

console.debug(17 === "17");//false

**第二个不同**:

在逻辑运算中，**0、""、false、null、undefined、NaN均表示false**。

----------------------------------------------------------------------------------

a && b : 将a, b转换为Boolean类型, 再执行逻辑与, 若表达式结果为true返回b, false返回a

a || b : 将a, b转换为Boolean类型, 再执行逻辑或, 若表达式结果为true返回a, false返回b

---------------------------------------------------------------------------------

&& 和|| 运算符的区别(开关)：

**&&操作:返回最后一个为true的值,或者第一个为false的值.**

**||操作:返回第一个为true的值,或则最后一个为false的值.**

console.debug(true && true)

console.debug(1 && true)

console.debug(1 && 2)

console.debug("A" && 2)

console.debug("" && 2)

console.debug(null && "B")

console.debug("A" && "B")

console.debug(1 && 2 && 3)

console.debug(1 && null && 3)

console.debug("" && null && 0)

if(123){

alert('OK');

}

if(0){

alert('OK');

}

## 函数(function)

### 函数的作用

1):将脚本编写为函数，就可以避免页面载入时执行该脚本。

2):函数包含着一些代码，这些代码可以多次被调用。

### 函数的定义和调用:

定义函数分类:普通函数/匿名函数.

函数必须调用才会执行.

定义语法:

function *函数名*(参数列表)

{

函数体

[return 值/变量]

}

### 函数的返回:

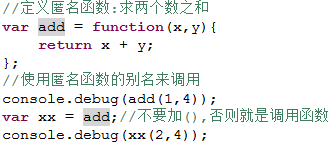
如果在函数中使用了return,那么函数的返回结果就是 return 后的**值**.

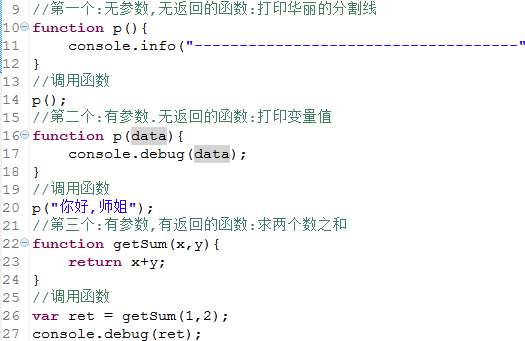
如果在函数中没有使用return,说明该函数没有结果,不用接受,若接受,结果为:undefined.

### 匿名函数(经常使用):

在定义函数的时候,没有定义函数名称.

此时,可以把函数当做是一个**值/变量,可以赋给另一个变量**.





## 全局变量和局部变量

**变量根据在function的内外定义位置不同,分成**:

**全局变量:function外部定义的变量称为全局变量**

**局部变量:function内部定义的变量称为局部变量**

访问变量原则：就近原则，谁离我最近我就使用谁。

-----------------------------------------------------

<script language="javascript">

var msg = "全局变量";

function show()

{

msg = "局部变量";

}

show();

alert(msg);//局部变量

</script>

-----------------------------------------------------

<script language="javascript">

var msg = "全局变量";

function show()

{

**var msg**;

msg = "局部变量";

}

show();

alert(msg);//全局变量

</script>

## 系统函数

**参见与:W3C javascript.chm ---> JavaScript 全局对象参考手册**.

在JS中已经预先定义好的,可以直接拿来使用的函数.

-------------------------------------------------------------

encodeURI及encodeURIComponent方法。

返回对一个URI字符串编码后的结果。

decodeURI及decodeURIComponent()方法。

将一个已编码的URI字符串解码成最初始的字符串并返回。

parseInt方法

将一个字符串按指定的进制转换成一个整数，语法格式为：parseInt(numString, [radix])。如果没有指定第二个参数，则前缀为 ‘0x’ 的字符串被视为十六进制，前缀为 ‘0’ 的字符串被视为八进制，所有其他字符串都被视为是十进制。

parseFloat方法

将一个字符串转换成对应的小数。

isNaN方法

检查一个值是否为NaN。

escape方法(不推荐使用,改为encodeURIComponent)

返回对一个字符串进行编码后的结果字符串。所有空格、标点、重音符号以及任何其他非 ASCII 字符都用 %xx 编码替换，其中xx等于表示该字符的Unicode编码的十六进制数，字符值大于255的字符以%uxxxx格式存储。

unescape 方法(不推荐使用，改为decodeURIComponent)

将一个用escape方法编码的结果字符串解码成原始字符串并返回。

eval 方法 \*

将参数字符串作为一个JavaScript表达式执行。



## 面向对象

对象中所包含的变量就是对象的属性，对象中所包含的对属性进行操作的函数就是对象的方法，对象的属性和方法都叫对象的成员。

类是对某一类事物的描述，是抽象上的概念；而对象实例是一类事物中的具体个例。

能够被用来创建对象实例的函数就叫对象的构造函数。使用new关键字和对象的构造函数就可以创建对象实例，

语法格式如下：var objInstance = new ObjName([传递给该对象的实际参数列表]);

----------------------------------------------

函数:面向过程的,可以独立存在和运行.

方法:面向对象的,函数在面向对象中的称谓,此时不能独立存在,因为要依附于对象,也使用对象来调用.

构造函数:专门用来构建对象的.

类:就是一个函数,是一个特殊的函数,构造函数.

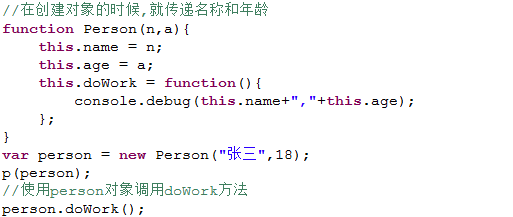
**js创建一个类,只需要定义类的构造函数(方法)**:



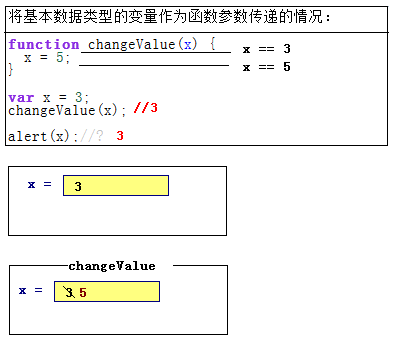
**在JS中this表示什么**? this就表示当前对象

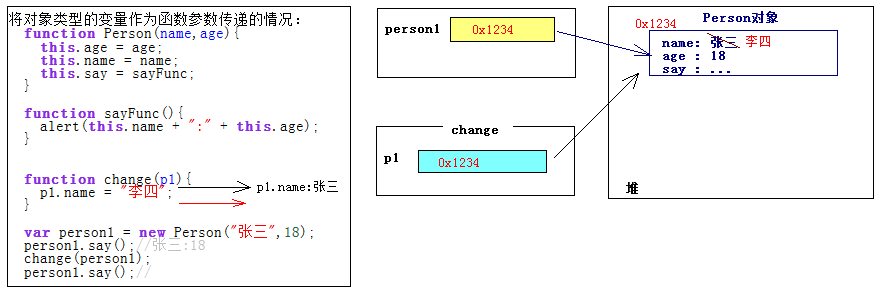
1):this在构造器(类)中,此时this就表示当前被创建的对象.

2):this在方法中,此时哪一个对象调用this所在的方法,this就是哪一个对象.



## 函数参数的值传递方式





## 内置对象

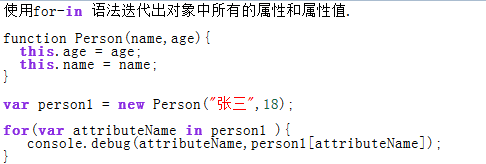
**for in循环好比是Java中的foreach循环**:

for in循环作用:

1.遍历数组里所有的数组元素的索引.

2.遍历js对象的所有属性.

--------------------------------------------------



**参见与:W3C javascript.chm ---> ECMAScript 引用类型**.

Object:

创建对象,并设置属性和方法

var obj = new Object();

obj.name = "will";

obj.age = 17;

obj.sayHello = function() {

};

// 对象的构造函数

alert(obj.constructor);

// 是否有指定的属性

debug(obj.hasOwnProperty("name1"));

// 迭代对象所有的属性+方法:for...in

for (attr in obj) {

alert(attr)//属性名

}

---------------------------------------------------------

Array(单独讲)

Boolean

----------------------------------------------------------

Date:

打印当前系统时间:

var d = new Date();

var day = d.getDate();

day = day<10?"0"+day:day;

var time = d.getFullYear() + "-" + (d.getMonth()+1) + "-" + day + " "

+ d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds();

----------------------------------------------------------

Math:

var num = Math.random();

----------------------------------------------------------

Number

----------------------------------------------------------

String:

// 随机生成A到Z之间的字母:65+26

// 随机生成0~25

var num = parseInt(Math.random() \* 26);//

num = num + 65;

alert(String.fromCharCode(num));

-----------------------------------------------------------

RegExp

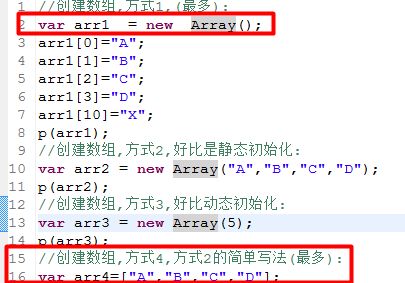


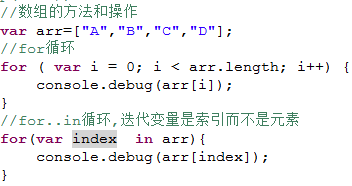
## 数组

### 为什么使用数组:存储多个数据.

js中的数组和Java中的ArrayList有异曲同工之妙,爽.

### 如何创建数组对象:





**操作数组的属性和方法**:

length－获得数组的长度；

concat－连接数组；

join－把数组转换成字符串；

pop－删除最后一个元素；

push－放入一个元素；

reverse－颠倒数据中的元素顺序；

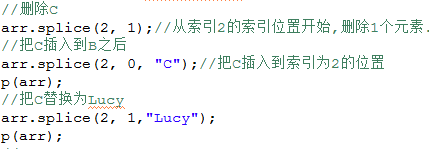
shift－移出第一个元素；

splice－截取数组;

sort－排序数组;

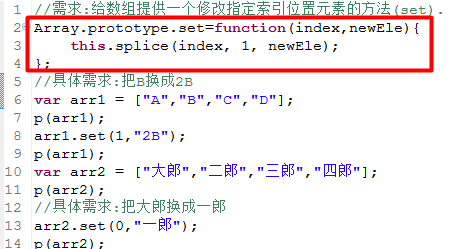
unshift－在前面追加元素；

**splice(start, deletecount, items) 方法用于插入、删除或替换数组的元素。**



## prototype

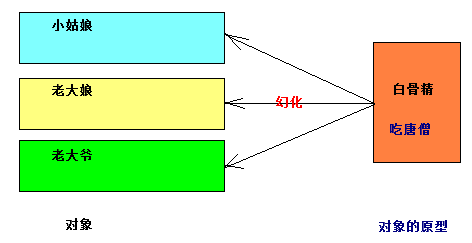
**需求:给数组提供一个修改指定索引位置元素的方法(set)**.



**使用prototype**:

**可以理解为在类上增加静态方法.**

**静态方法属于类,属于该类所有的对象,如此该方法就属于所有的对象.**

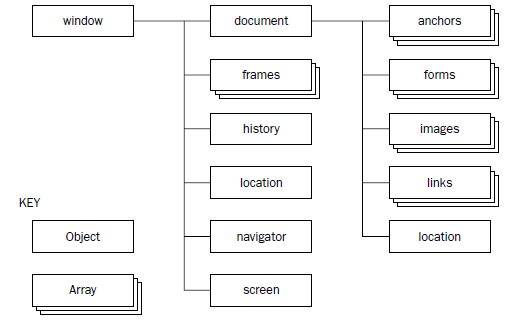
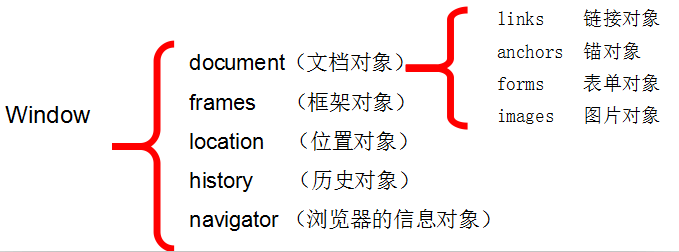


小结



## BOM和Document对象常见属性和方法

**BOM是browser object model的缩写，简称浏览器对象模型**.



**Document对象常见属性和方法**:

wondow.onload事件:当页面加载完成做一些事情

-----------------------------------------------

links 链接对象

anchors 锚对象

forms 表单对象

images 图片对象

------------------------------------------------

document对象：能反映当前页面的各种属性。

**document对象的常用属性**

title:显示文章的标题。

body：获取body元素节点。

bgColor:显示页面的背景色。

fgColor:显示页面的前景色。

document对象的常用方法：

write():在页面中输出文本。

writeln():在页面中输出文本，并自动换行(多一个换行符,看不出来)。

------------------------------------------------

**访问表单的方式**

document.forms[索引]:获取当前页面中的第几个表单(索引从0开始).

document.表单名称:

document.getElementById("表单的id值");

Document 对象

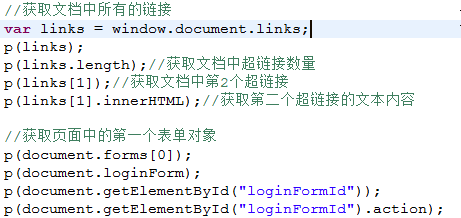
每个载入浏览器的 HTML 文档都会成为 Document 对象。

Document 对象使我们可以从脚本中对 HTML 页面中的所有元素进行访问。

window.onload = function(){

//处理文档加载完之后的操作.

};



## 消息框/输入框/确认框

alert(‘信息’)：显示带有一段消息和一个确认按钮的警告框

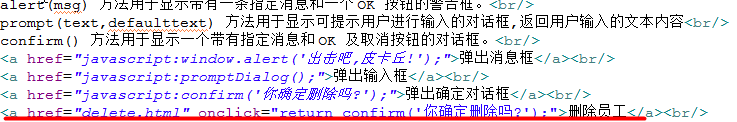
//alert("你好1");

prompt(‘提示信息’,默认值)： 标准输入框

//var inputText = prompt("请输入你的名字","孙悟空");

confirm( ) ： 确认框

//var ret = confirm("亲,你确定删除吗?");



## 打开新窗口/关闭窗口

window.open()方法：打开一个新窗口

调用示例：

window.open("http://www.520it.com/", "\_blank","")

第一个参数：要打开的网页url，可以是相对路径;

第二个参数：打开窗口的目标；除了自定义名称以外，还包括\_self, \_parent, \_top及\_blank几个特殊值;

第三个参数：是一个使用,组成的字符串，用于描述打开窗口的特性，比如大小、是否有工具栏等。

function openWin() {

open("newWin.html");

}

close( )：关闭窗口

function closeWin() {

close();

}

在Firefox中:close方法只能关闭使用js的open方法打开的窗口.

## 窗口大小位置设置

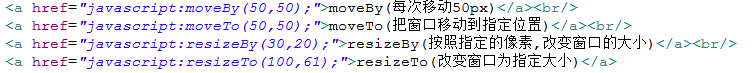
**IE中有效**

// moveBy(dx, dy)-移动偏移量

// moveTo(x, y)-移动到指定座标

// resizeBy(dw, dh)-窗口大小改变

// resizeTo(w, h)-窗口大小改变



## 浏览器位置(IE兼容性)

获得窗口在屏幕上的位置

IE:

screenLeft -获得窗口所在X坐标值

screenTop -获得窗口Y坐标值

W3C:

screenX -获得窗口所在X坐标值

screenY -获得窗口Y坐标值

---------------------------------------------------------

function getWindowSize() {

var x = window.screenX || window.screenLeft || 0;

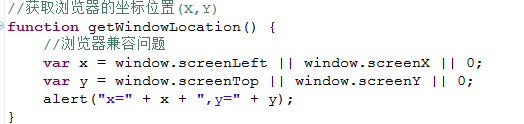
var y = window.screenY || window.screenTop || 0;

alert("X坐标=" + x + ", Y坐标=" + y);

}

判断是否是IE浏览器





## 访问历史

**查看浏览器历史**:

history.go(index)函数，在浏览器历史记录中跳转，正数为前跳，负数为后跳

history.back()函数，后跳;

history.forward()函数，前跳;

history.length属性，获得历史记录的长度；



## 浏览器相关信息

navigator对象用于获得浏览器相关信息;使用console.dir(navigator)查看成员信息。

属性名称 说明

appCodeName 产品名称

appName 应用名称

appVersion 版本号

cookieEnabled 是否允许使用cookie

language 语言

oscpu 操作系统名称

platform 操作系统平台

product 产品代号或名称，比如Gecko

productSub 产品发布日期20100701

userAgent 客户端详细信息,比如：Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.11) Gecko/20100701 Firefox/3.5.1

--------------------------------------------------------------------------------

判断操作系统:

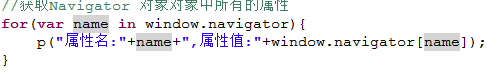
var isWin = (navigator.platform == “Win32”) || (navigator.platform == “Windows”);

var isMac = (navigator.platform == “Mac68K”) || (navigator.platform == “MacPPC”)

|| (navigator.platform == “Macintosh”);

var isUnix = (navigator.platform == “X11”) && !isWin && !isMac;





## 访问页面的url

window对象的location属性,保护了该窗口所装载文档的地址:location包含的常用属性:

1.hostname:文档所在地址的主机名

**2.href:文档所在地址的URL地址**

3.host:wend所在地址的主机地址

4.port:文档所在地址的服务端端口

5.pathname:文档所在地址的文件地址

6.protocol:装载该文档使用的协议

----------------------------------------------------------

var loc = window.location;

for(var name in loc){

console.debug(name,loc[name]);

}

表示直接:修改浏览器的地址栏为:http://jd.com

<a href="javascript:window.location.href='http://jd.com'">点击之后,跳转到[京东商城]</a>



## 定时器

setTimeout(fn,time)函数，返回timer，用于指定一段时间后执行某函数

setInterval(fn,time)函数，返回timer，用于周期性执行某函数;

参数fn可以是字符串组成的jascript代码，也可以是一个函数名称；

参数time表示时间，毫秒为单位；

clearTimeout(timer)函数，清除由setTimeout创建的定时器；

clearInterval(timer)函数，清除由setInterval指定的定时器；

**电子时钟**:

2015-10-27 11:58:34



## DOM和DOM和获取元素的三种方式

DOM ：DOM是Document Object Model**文档对象模型**的缩写。根据W3C DOM规范,DOM是一种与浏览器，平台，语言无关的接口，使得你可以访问页面其他的标准组件.

D：文档 – html 文档 或 xml 文档

O：对象 – document 对象的属性和方法

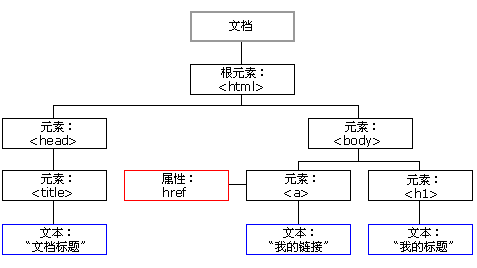
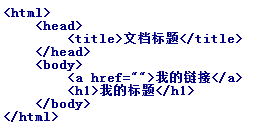
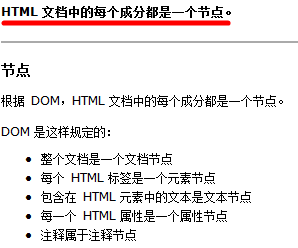
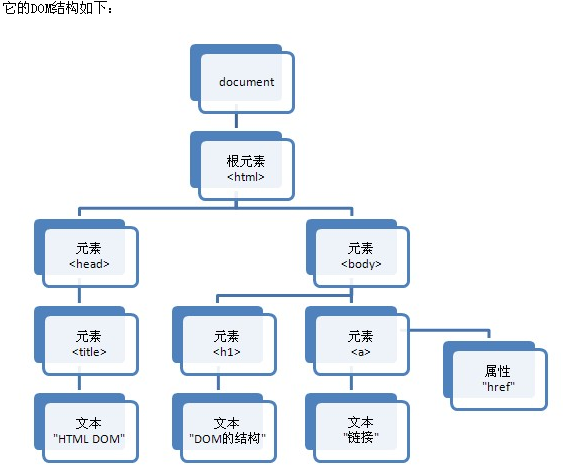
M：模型

DOM 是针对xml(html)的基于树的API。

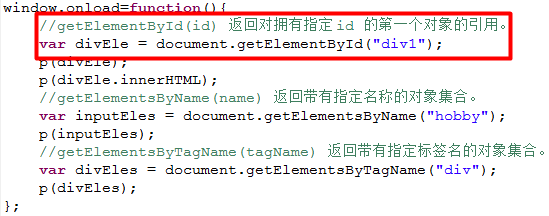
DOM树:节点（node）的层次。

DOM 把一个文档表示为一棵家谱树（父，子，兄弟）

DOM定义了Node的接口以及许多种节点类型来表示XML节点的多个方面

**DOM和获取元素的三种方式**:



## Node对象的属性和方法



**Node对象常用属性和方法**:

firstChild

lastChild

childNodes

previousSibling

nextSibling

asChildNodes()

**元素节点/文本节点/属性节点的nodeName/nodeValue/nodeType区别**:

noteType nodeName nodeValue

元素节点 1 当前元素标签名 null

属性节点 2 当前属性名称 当前属性的值

文本节点 3 #text 当前文本内容



## 属性操作

**属性操作**:

1):给某个元素上的某个属性设置值.

2):获取某个元素上的某个属性的值.

-------------------------------------------

1.元素上的原始属性和自定义属性的获取值和设置值

操作元素的原始属性:

**获取属性值**:

元素对象.属性名;

元素对象["属性名"];

**设置属性值**:

元素对象.属性名=值;

元素对象["属性名"]=值;

操作元素**自定义属性**:

获取属性值:

元素对象.getAttribute("属性名");

设置属性值:

元素对象.setAttribute("属性名",值);

2.操作属性名和默认属性值相同的属性. 如:checked,selected

在DOM中的:属性值为true(选中)和false(不被选中).

3.操作class属性.-->属性名为:className

在html中属性为class,在DOM中对应className.

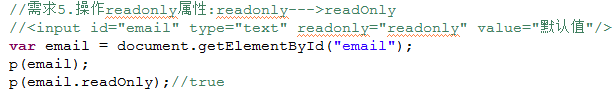
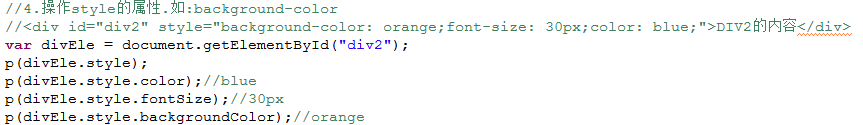
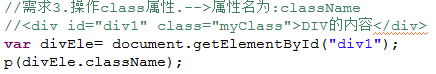
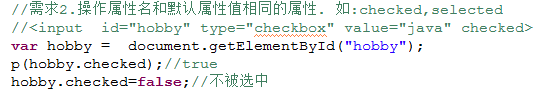
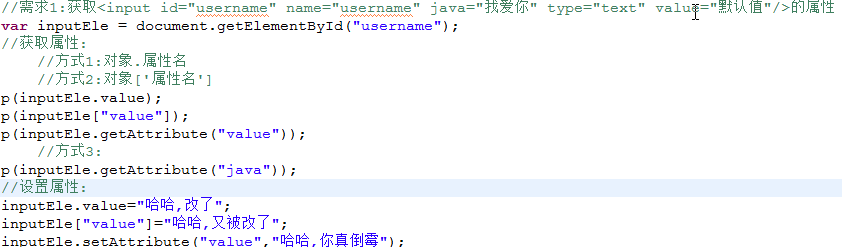
4.操作style的属性.如:background-color

在html中的属性:background-color,在DOM中对应:

backgroundColor.

5.操作readonly属性:readonly--->readOnly

在html中的顺序:readonly,在DOM中:read**O**nly;



## 插入/追加/删除/替换节点

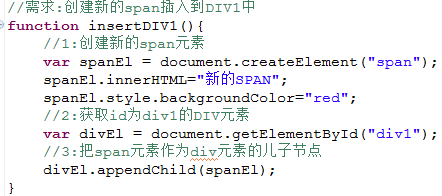
**插入/追加/删除/替换节点**:

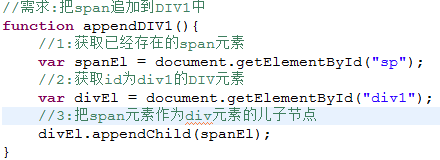
1.给父节点追加现有/新子节点

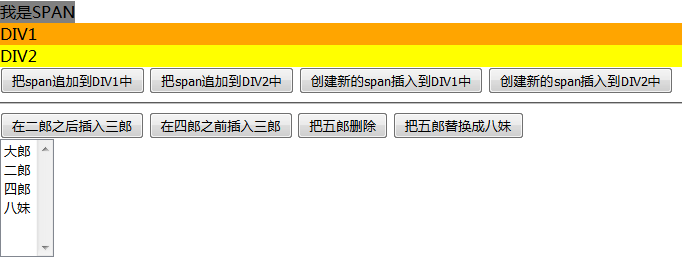
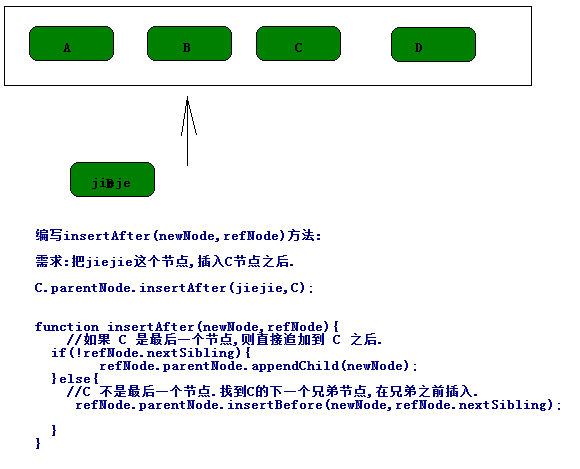
2.在指定节点之前/之后插入新节点

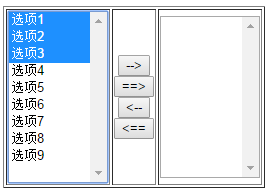
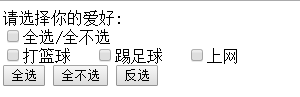
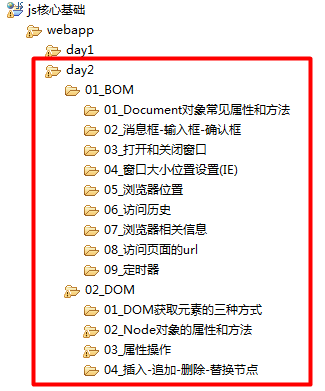
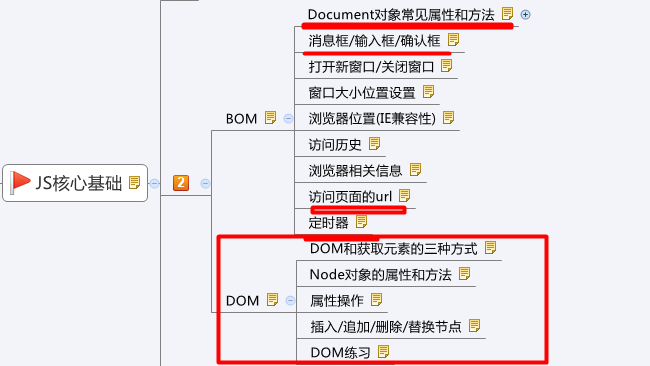
3.删除节点/替换节点









小结

## DOM事件编程

**事件驱动编程**:

所谓事件驱动,简单地说就是你点什么按钮(即产生什么事件),电脑执行什么操作(即调用什么函数).当然事件不仅限于用户的操作. 当对象处于某种状态时，可以发出一个消息通知，然后对这个消息感兴趣的程序就可以执行。

**事件驱动编程中的几个核心对象**:

1):事件源：谁发出事件通知，发出消息；也就是事件主体（通常指元素和标签）；

2):事件名称：发出什么样的通知的名称，比如鼠标到我头上了，我被别人点了一下；

3):事件响应函数：谁对这个这个事件感兴趣，当这个事件发生时要执行什么样的操作；

4):事件对象：一般来说，当事件发生时，会产生一个描述该事件的具体对象，包括具体的参数等一起发给响应函数，好让他们通过事件对象来了解事件更加详细的信息。

**很美丽动人的事故(事件):**

**有一天,在街上发现一个美女,小明伸出手在美女的脸蛋点击了一下,美女回头给了一巴掌,很痛!**

**事件源**: 美女.

**事件名称**: 点击.

**事件响应**: 给了小明一巴掌.

**事件对象**: 时间,地点,人物,事情....

------------------------------------------------------------------------------

**示例**:

<p style="color: red" id="p1">Hello world!</p>//事件源

function shout(**e**){//响应函数，监听函数

alert(e.clientX);//e事件对象

}

document.getElementById("p1").onclick=shout;//在事件源上绑定事件响应函数

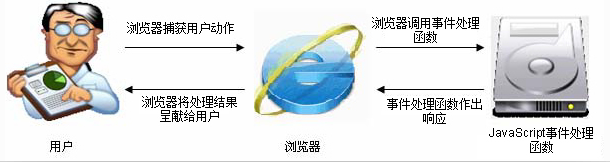
**JavaScript处理事件的基本机制**：

1):对DOM元素绑定事件处理函数；

2):监听用户的操作；

3):当用户在相应的DOM元素上进行与绑定事件对应的操作时，事件处理函数做出响应；

4):将处理结果更新到 HTML 文档。



## 如何给HTML中的元素(事件源),绑定事件(有三种方式):

### 方式1:

直接在元素上设置**onXxx属性**.(xxx表示事件名称).

<p onclick="*调用响应函数*">P标签1</p>

使用JS内置的函数:

<p onclick="alert(123);">P标签1</p>

使用用户自定义的函数:

<p onclick="***sayHello(this);***">P标签1</p>

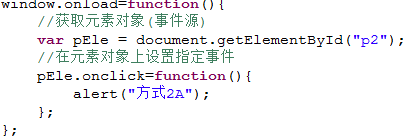


方式1:很简单也很直观,但是同一种的事件只能绑定一次.

**方式2**:

在元素对象上直接设置onXxx事件属性,再给事件属性设置响应的函数.

<p id="p2">P标签2</p>



对应同一种事件,后绑定的事件响应函数会覆盖之前该事件绑定的响应函数.

**方式3**:

使用方法来完成对元素的监听.

**IE浏览器**:

[Object].attachEvent(“name\_of\_event\_handler”, fnHandler);

name\_of\_event\_handler>>**事件操作前,必须加"on"!!!**

多次添加监听后,触发顺序: 先添加,后执行

[Object].detachEvent(“name\_of\_event\_handler”, fnHandler);

fnHandler : 移除时,传入的"事件响应函数",必须和添加时,传入的是同一个(通过相同标识符引用的那一个函数)

匿名函数,每次创建的都不同

**W3C浏览器**:

[Object].addEventListener(“name\_of\_event”, fnHandler);

name\_of\_event>>直接使用事件(操作)名称

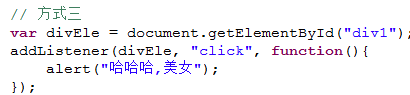
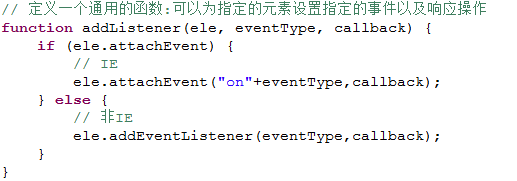
多次添加监听后,触发顺序: 先添加,先执行

IE浏览器从IE9开始,也支持.

[Object].removeEventListener(“name\_of\_event”, fnHandler);

fnHandler : 移除时,传入的"事件响应函数",必须和添加时,传入的是同一个(通过相同标识符引用的那一个函数)

匿名函数,每次创建的都不同



**事件其他相关的信息**:



**事件对象中的属性**:





## 打字游戏

**打字游戏步骤**:

1.准备一个游戏界面,提供开始游戏按钮

禁用按钮,在页面加载完毕,即时启用"开始游戏"按钮

2.对开始按钮,添加点击事件

2.1) 定时随机生成字符,并显示到页面中

2.1.1> 创建<lable>元素,并设置随机字符

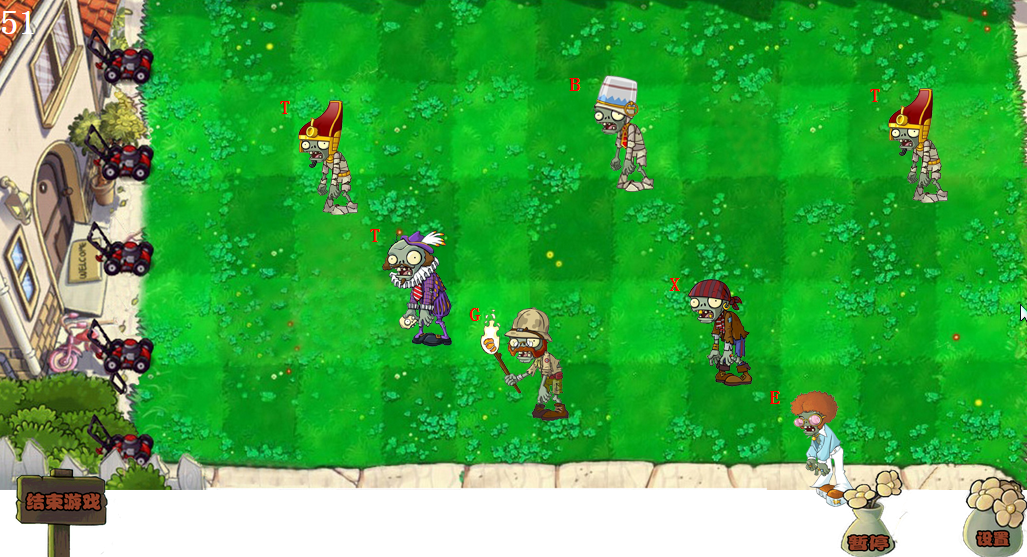
<label style="position:absolute;top:30px;left:10px;background-color:red;width:20px;text-align:center;font-weight:bold;">I</label>

2.1.2> 把<lable>元素添加到body元素中

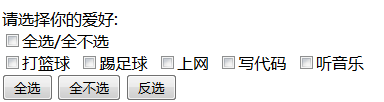
2.2) 让页面中的字符周期性向下移动

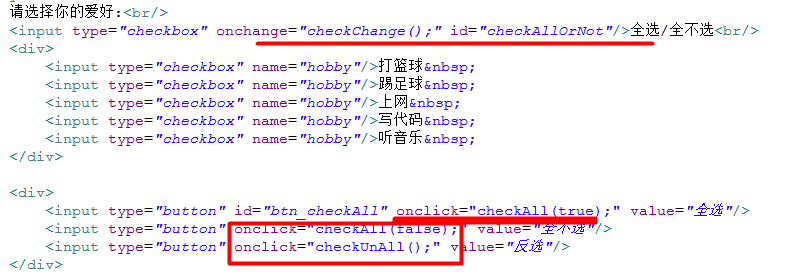
2.3) 对键盘输入内容,监听,对比输入内容,和游戏界面字母内容,一致则消除

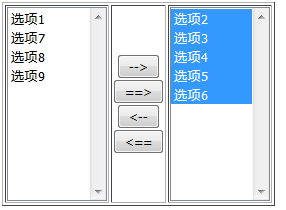


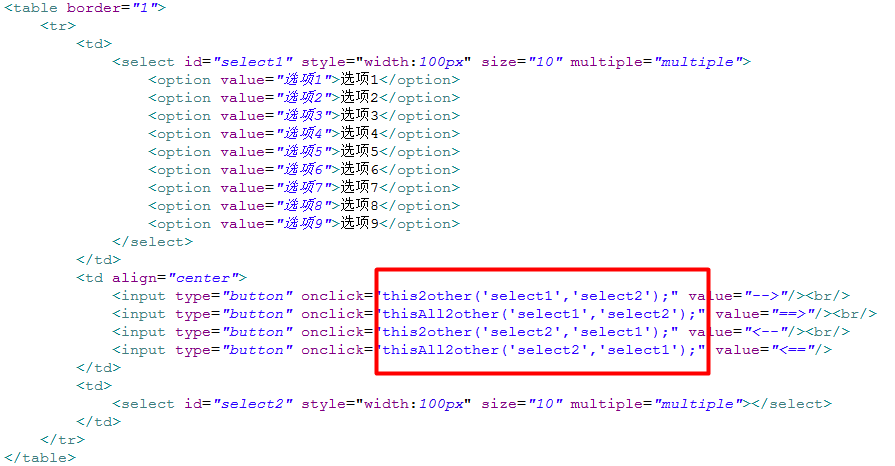


## 练习题







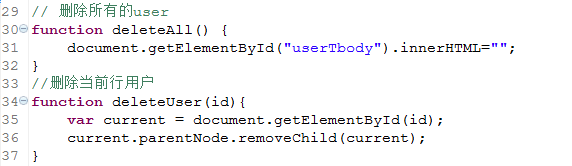




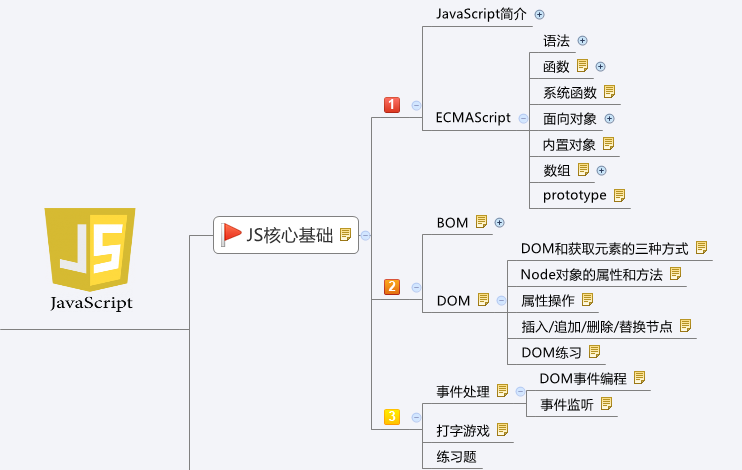








小结



# Ajax概述和实用需求

**Ajax介绍/阿贾克斯**:

一、Ajax不是一项具体的技术，而是几门技术的综合应用。

Javascript、XHTML和CSS、DOM、XML和XMLHttpRequest。

二、Ajax核心只不过是要在javascript中调用一个叫**XMLHttpRequest类**，这个类可以与Web服务器使用HTTP协议进行交互，程序不通过浏览器发出请求，而是用这个特殊的JavaScript对象发送请求和接收响应。

三、XMLHttpRequest对象在网络上的俗称为**Ajax对象**。

**四、一种不用刷新整个页面便可与服务器通讯的办法(更新网页部分数据)**.

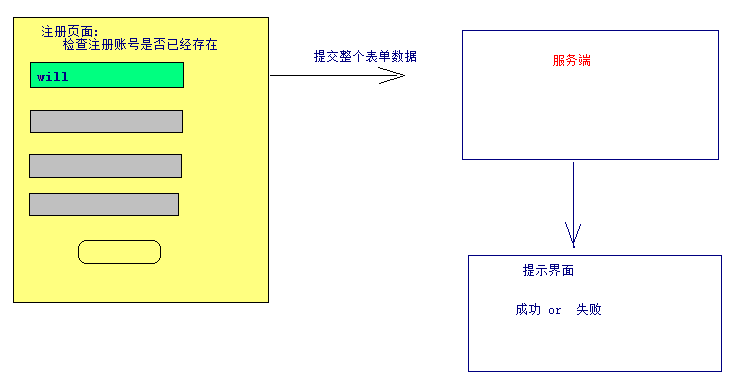
-------------------------------------------------------------------------

AJAX 指**异步 JavaScript 及 XML（Asynchronous JavaScript And XML）**。

AJAX 是一种在 2005 年由 Google 推广开来的编程模式。

AJAX 不是一种新的编程语言，而是一种使用现有标准的新方法。

通过 AJAX，你可以创建更好、更快以及更友好的 WEB 应用程序。



**Ajax的特点(看图)**：

浏览器中显示一个页面后，这个页面以后一直不改变，所有的操作请求都由这个网页中的javascript代码发出，所有的结果都由javascript代码接受并增加到这个页面上，浏览器窗口中显示的网页始终都是初始的那个网页。

增强用户体验：可以在用户浏览网页的同时与服务器进行**异步交互**和实现网页内容的局部更新。

**同步和异步交互**：

\* **同步**：提交请求->**等待服务**

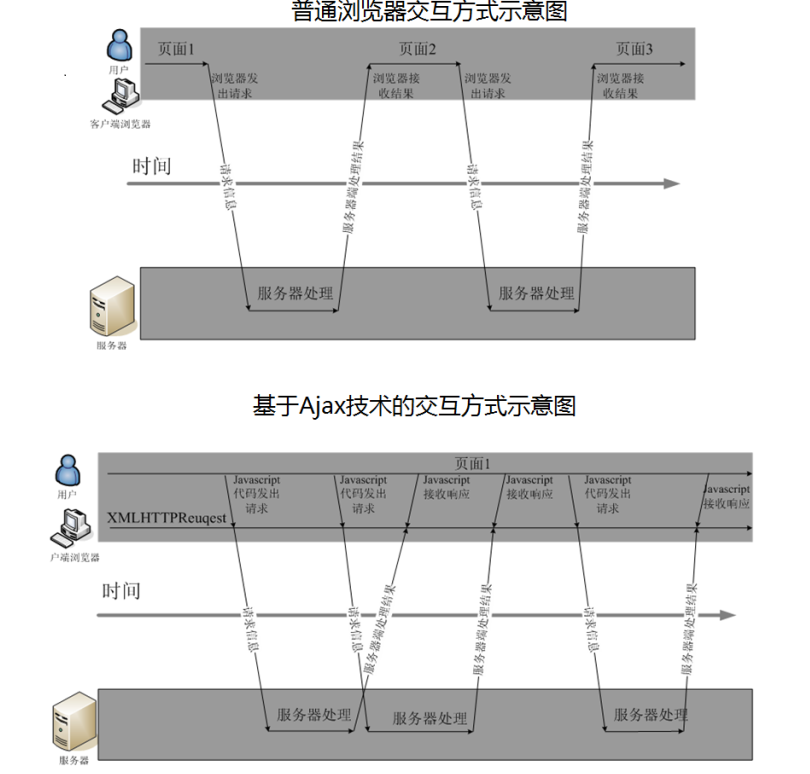
**器处理**->处理完毕返回 这个期间客户端**浏览器不能干任何事(单线程)**.

\* **异步**：请求通过事件触发->服务器处理（这时浏览器仍然可以作其他事情）->处理完毕(**多线程**).

同步是指：发送方发出数据后，等接收方发回响应以后才发下一个数据包的通讯方式。

异步是指：发送方发出数据后，不等接收方发回响应，接着发送下个数据包的通讯方式。

**具体应用**: **NBA文字直播平台**.



**AJAX缺陷**：

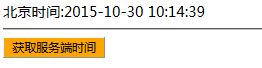
1，AJAX大量使用了Javascript和AJAX引擎，而这个取决于浏览器的支持。IE5.0及以上、Mozilla1.0、NetScape7及以上版本才支持，Mozilla虽然也支持AJAX，但是提供XMLHttpRequest的方式不一样。所以，使用AJAX的程序必须测试针对各个浏览器的兼容性。

2，AJAX更新页面内容的时候**并没有刷新整个页面，因此，网页的后退功能是失效的**；有的用户还经常搞不清楚现在的数据是旧的还是已经更新过的。这个就需要在明显位置提醒用户“数据已更新”。

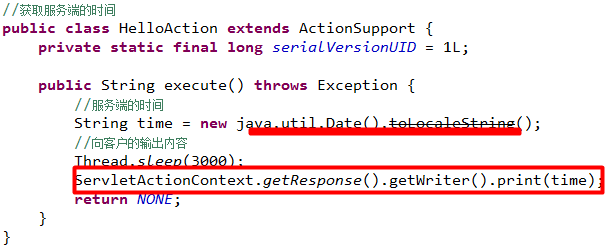
3，对流媒体的支持没有FLASH、Java Applet好。

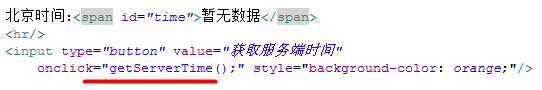
::::**html5 + Ajax(jQuery) 取代 Flash**.

## Ajax简单入门:获取服务端时间



**AJAX不支持跨域访问**:







## 浏览器兼容和缓存问题

**Ajax的问题**:

**1):只接受和处理响应成功**.

**2):XMLHttpRequest针对于W3C和IE7以上版本的浏览器有效**.

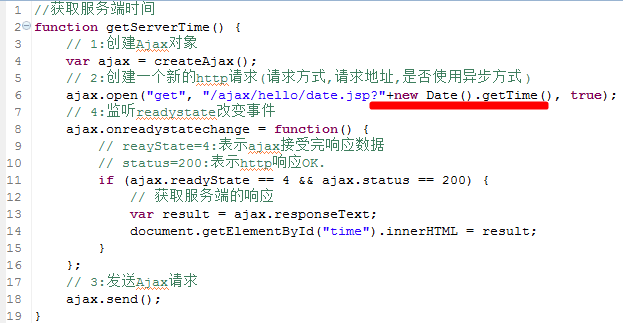
针对于IE6和IE6以下的版本不支持.

3):**IE浏览器默认会对Ajax请求缓存**.

把请求的URL当做Map中的key,把该请求的响应作为Map的value.

如果请求地址不变,就直接把上一次该地址的响应给客户端(不会发生新的请求).

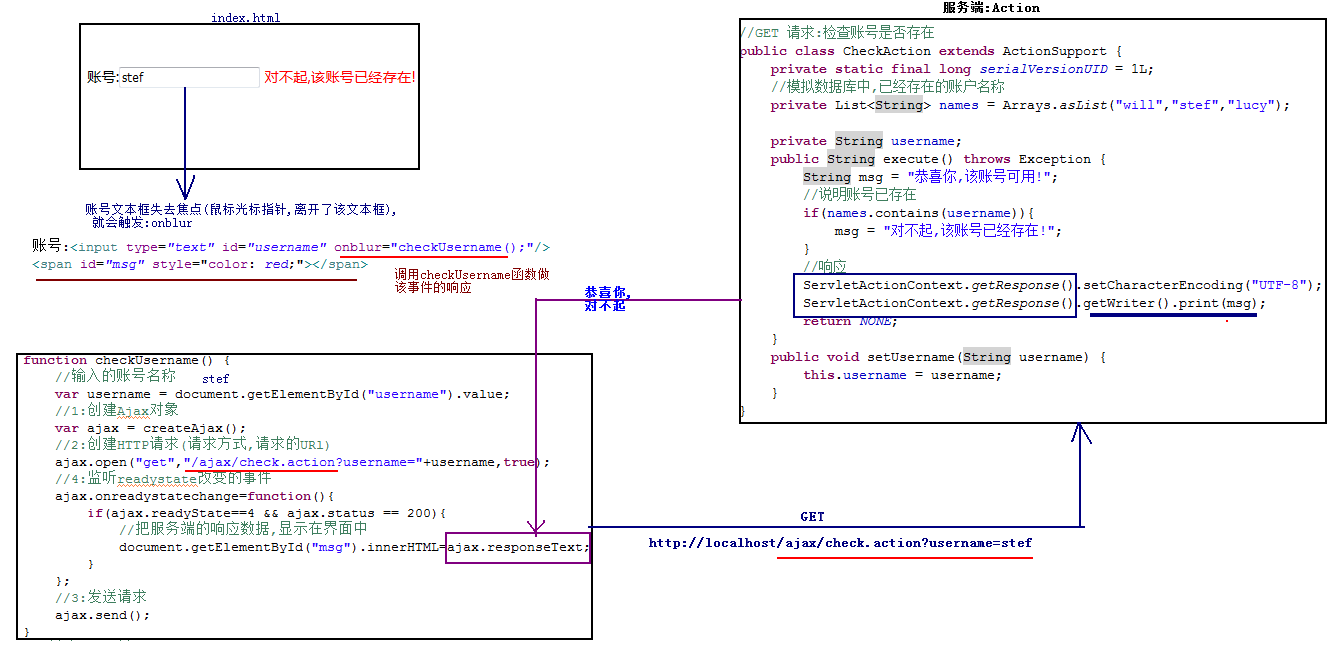
**解决方案:使用时间戳作为资源的参数**.



## GET 请求:检查账号是否存在

**使用Ajax发生GET请求,并传递参数**.

**案例:检查注册账号是否已经存在**.



## POST请求:登录操作

**使用Ajax发生POST请求,并传递参数**.

**案例:使用Ajax完成登陆操作**.



## 二级联动

就拿你在网上注册的时候要填个人地址,

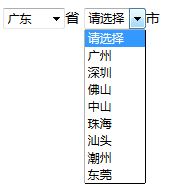
**我假设它有两个下拉列表(其实还可以有更多)**

第一个下拉列表中让你选择的省,而另一个下拉列表让你选择的是城市,当你在省的下拉列表中的选择发生改变的时候,城市的下拉列表也应当跟着你所选择的省名称而发生改变.这样就产生了一种联动的较果.

也就是简单的二级联动..



**使用html作为后台和前台的数据交互格式**.



**使用xml作为后台和前台的数据交互格式**.

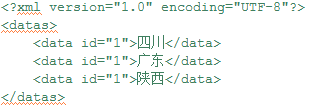
--------------------------------------------------------------------

在服务端中:把所有的数据以**XML文档**的结构封装起来,再响应给客户端.

在Ajax处理响应的时候: 解析响应过来的XML文档,获取其中的数据,再拼接在<option>元素中,最后把<option>元素作为<select>的儿子节点.

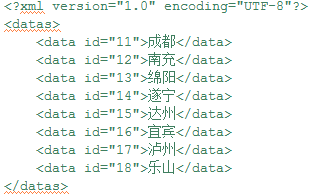


**省份的XML数据格式**:

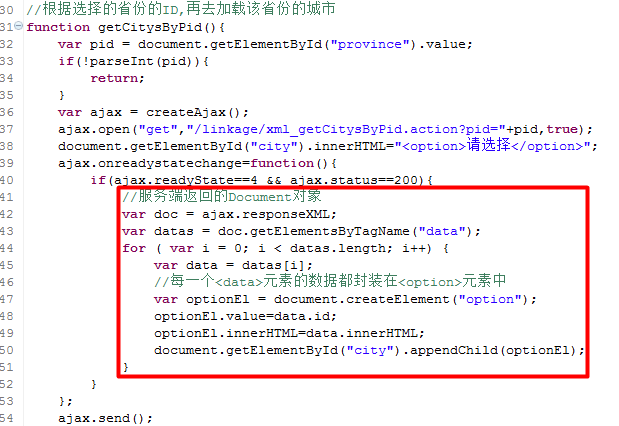




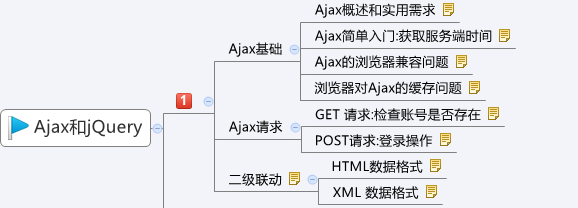
**城市的XML数据格式**:







小结



**熟练完成Ajax的知识点**:

-----------------------------------------------

自行预习:

--->JSON:

--->jQuery:

# JSON数据格式

JSON（JavaScript Object Notation）一种简单的数据格式，比xml更轻巧。易于人阅读和编写，同时也易于机器解析和生成(网络传输速度快)

JSON是JavaScript原生格式，这意味着在JavaScript中处理JSON数据不需要任何特殊的API或工具包。

描述一个对象 {属性名:属性值,属性名:属性值}

JSON的规则很简单：

对象是一个无序的“‘名称/值’对”集合。一个对象以“{”（左括号）开始，“}”（右括号）结束。每个“名称”后跟一个“:”（冒号）；“‘名称/值’对”之间使用“,”（逗号）分隔。

规则如下:

1）映射用冒号（“：”）表示。名称:值

2）并列的数据之间用逗号（“，”）分隔。名称1:值1,名称2:值2

3） 映射的集合（对象）用大括号（“{}”）表示。{名称1:值1,名称2:值2}

4） 并列数据的集合（数组）用方括号(“[]”)表示。

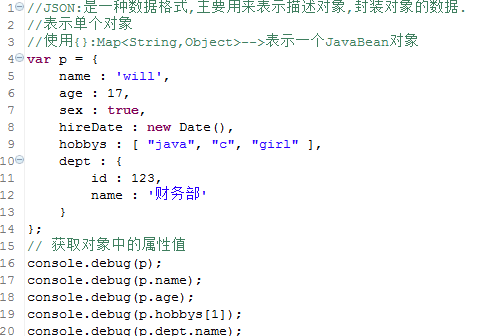
[

{名称1:值,名称2:值2},

{名称1:值,名称2:值2}

]

5 元素值可具有的类型：string, number, object, array, true, false, null



**注意:**

1.JSON数据格式:(注意:标准的JSON的属性也使用引号)

2.字符串转成对象:

3.json对象的toSource方法

------------------------------------------------------------

var str = "{name : 'will',age : 17}";

console.debug(str);

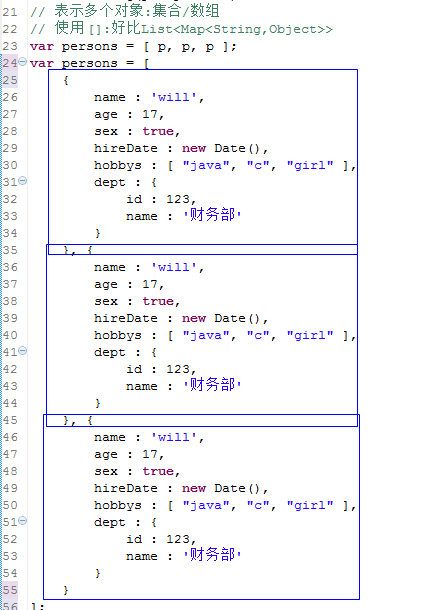
//eval函数,可以把一个字符串转换为JS代码

var p = eval("("+str+")");

console.debug(p.toSource());

------------------------------------------------------------

最好是:在JSON字符串前后,都带有();



## 使用JSON实现二级联动

**使用JSON作为后台和前台的数据交互格式**.

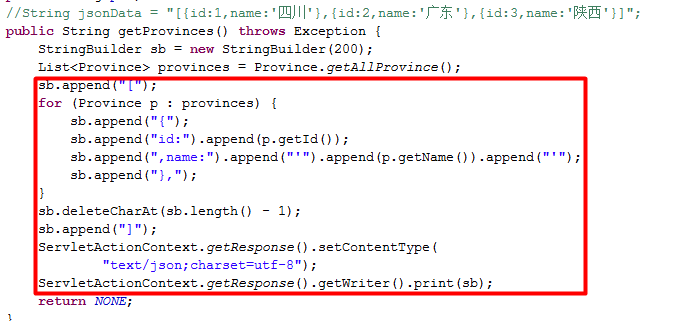
--------------------------------------------------------------------

在服务端需要把省份/城市的数据,**拼接成一个JSON的字符串,并响应给JS**.

JS需要把字符串的JSON数据,转换为JSON对象.-->每一个对象,都创建一个<option>元素,并存放数据.

**使用JSON表示省份**:

String jsonData = "**[{id:1,name:'四川'},{id:2,name:'广东'},{id:3,name:'陕西'}]**";



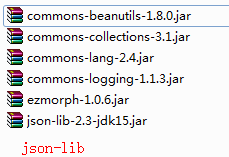


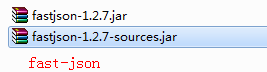
## JSON转换工具(JSON lib和fast JSON)

**JSON的转换工具(提供JSON对象和Java对象(JavaBean/集合)的相互转换方法)**:

1):JSON lib.(最广泛的,需要依赖另外的5个jar).

2):fast json(阿里开发的,号称世界第一,不需要依赖,有一个即可.)





## Struts2的JSON插件

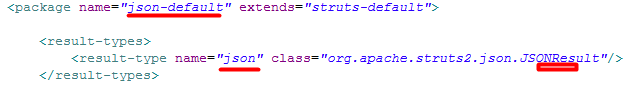
**Struts2本身是支持AJAX操作的,支持把对象转换为JSON数据格式**.

**但是依赖插件**:

struts2-json-plugin-2.3.24.jar

----------------------------------------------------------------------------

**提供了json的结果类型**:



**默认,可以把值栈最顶端(栈顶)的Action中属性的数据解析成JSON字符串**.



# jQuery简介

**1，jQuery简单背景介绍**；

jQuery是继**prototype**之后又一个优秀的Javascript框架,(**Extjs**)。

jQuery是轻量级的js库 ，它兼容CSS3，还兼容各种浏览器（IE 6.0+, FF 1.5+, Safari 2.0+, Opera 9.0+）

**jQuery是免费、开源的**。

jQuery是一个兼容多浏览器的javascript库，**核心理念是write less,do more(写得更少,做得更多).**

2，jQuery版本介绍；

jQuery1.x： 经典版本，兼容IE6，7，8；

jQuery2.x：改进版本，及后续版本将不再支持IE6，7，8浏览器。

jQuery能干什么？

jQuery使用户能更方便地处理HTML（标准通用标记语言下的一个应用）、events、实现动画效果，并且方便地为网站提供AJAX交互。

jQuery的语法设计可以使开发者更加便捷，例如操作文档对象、选择DOM元素、制作动画效果、事件处理、使用Ajax以及其他功能。

3，JQuery文件介绍；

jQuery1.x.js：jQuery源文件；学习jQuery或者debug的时候使用；

jQuery1.x.min.js：jQuery压缩之后的文件；正常项目中使用；



## jQuery的引入和体验

**jQuery引入(安装)**:

第1步:引入jQuery的库文件:<script type="text/javascript" src="../jquery-1.11.2.min.js"></script>

第2步:引入我们自己的js : <script type="text/javascript" src="index.js"></script>

第3步:在index.js中,alert($),如果输出的是一个函数,说明安装成功!

出错效果如下:

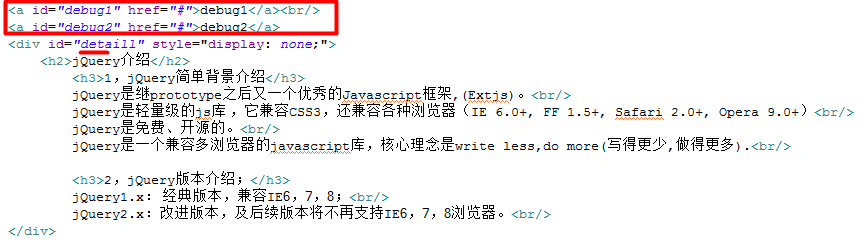
使用不同的jQuery插件,可能依赖的jQuery库的版本不一致,此时请保留插件依赖的jQuery库.

**神奇的$:**

在jQuery中,$其实就是jQuery对象($ === jQuery)**.**

**-------------------------------------------------------------------------------------------------------------------------------**

**案例:模拟Struts2的debu标签.**



**$(function(){**

**//页面加载完毕之后,就执行的代码,相当于window.onload事件**

**});**

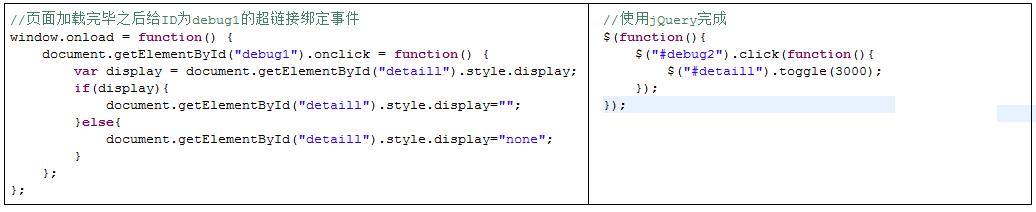
---------------------------------------------------------------------------

window.onload=function(){};只能编写一次,后写会覆盖之前写的.

**而$(function(){});可以 写N次**.

--------------------------------------------------

$("#**ox**"):获取ID属性为ox的元素对象.



## jQuery对象

**需求**:获取和修改某个元素的文本内容.

-----------------------------------------------------------------------------------

引入问题：通过document.getElementById()找到的console内容和通过$();找不到元素console返回的内容不一样；

1，**通过jQuery方法获取的页面元素，都是jQuery对象**。

2，jQuery对象其实就是对**DOM对象进行了包装**，增强相关了方法，让开发者使用起来更加便利。

jQuery对象其实是**模拟了数组(伪数组)**,**每一个索引对应一个真正的DOM元素**.

jQuery对象是不可能为null的,可以根据jQuery的length是否为0来判断,当前是否查询出了DOM元素.

3，虽然jQuery对象包装了DOM对象但是两种不能混用，各位可以理解为jQuery对象与DOM对象是两个不类型的对象，但是我们调用jQuery对象的方法，事实上底层代码还是操作的是DOM对象。

-----------------------------------------------------------------------------------------------------------------------

jQuery对象和DOM对象的关系,是jQuery对象包含了DOM对象.

**jQuery对象和DOM对象的获取方式,以及相互的转换问题**:

1):使用jQuery方法获取的对象都是jQuery对象.

2):使用document.getElementById/getElementsByTagName/getEelementsByName获取的都是DOM对象.

**jQuery对象转换为DOM对象:**

jQuery对象[index]或者jQuery对象.get(index):表示当前jQuery对象中包装的第index个DOM元素.

**DOM对象转换为jQuery对象**: **$(**DOM对象**)**;

----------------------------------------------------

**jQuery对象只能访问jQuery对象的属性和方法,不能访问DOM对象的属性和方法.**

**DOM对象只能访问DOM对象的属性和方法,不能访问jQuery对象的属性和方法.**

## jQuery选择器概述/如何学习jQuery/jQuery常用方法

**文档在手,天下我有**!

------------------------------------------------------------------------------------

**jQuery选择器是jQuery类库最重要功能之一**。

这些选择器的用法和CSS的语法非常相似，结合jQuery类库的方法你可以很方便快速地定位页面中任何元素,并为其添加响应的行为。

-------------------------------------------------

一、什么是选择器？

**jQuery提供获取页面元素一种语法**。

二、选择器的组成？

选择器一般由“特殊符号”+“字符串”组成。

比如:“#”代表id “mydiv”是一个字符串，那么整体#mydiv代表,id为mydiv的元素对象。

三、使用选择器获取元素?

$(“选择器”) ， $ (“#mydiv”)

注意：如果通过jQuery方法获取页面中元素，没有查找到，返回值不是null,返回值为一个空数组[]，所以判断是否获取到元素，通过jquery.length!=0来判断。

过滤选择器:**通过特定的过滤规则来筛选所需要的DOM元素**.过滤规则与CSS中的伪类选择器语法相同.

即选择器以一个冒号(:)开头.

按照不同的过滤规则,可分为基本过滤,内容过滤,可见性过滤,属性过滤,子元素过滤,表单对象属性过滤选择器.

## jQuery事件绑定

**js中绑定事件,三种方式**:

方式1: 直接在元素上,增加onXxx事件属性.

<button **onclick="alert(1);"**>点我</button>

方式2:获取元素对象,再设置onXxx事件属性.

<button  **id="btn"**>点我</button>

document.getElementById("btn").**onclick**=function(e){

//TODO

}

方式3:使用addEventListener/attachEvent

IE9之前: 只支持attachEvent("eventType",fn);

document.getElementById("btn").attachEvent("**on**click",function(){ TODO });

W3C/IE9之上: 支持addEventListener

document.getElementById("btn").addEventListener("click",function(){ TODO });

**文档加载完毕**:

window.onload=function(){

//文档加载完毕之后,就执行这里

};

**只能使用一次**.

-------------------------------------------------------------

**传统写法**:

$(document).ready(function(){

// TODO

});

**简写版本**:

$(function(){

// TODO

});

**可以使用多次**.

## 在jQuery中如何绑定事件:

通过bind函数:

$("#btn1").bind("click",function(){

alert('点击按钮1');

});

取消绑定事件:通过unbind函数.

$(":button").unbind();//把所有按钮上所有事件都取消绑定.

$(":button").unbind("click");//取消所有按钮上的click事件.

--------------------------------------------------------------------------------------------------------

**在jQuery中,为绑定事件提供了更简单的方式,其底层和上述相同**.

jQuery对象.xxx(fn);

$("#btn1").clkick(function(){

//TODO

});

**并且可以为同一个元素绑定多次相同的事件**.

## jQuery的DOM操作

看案例



## jQuery的Ajax请求

**传统的Ajax操作**:

1:创建Ajax对象(**XMLHttpRequest/ActiveXObject**);

2:开启一个请求.

ajax.open("get/**post**",url,**true**);

4:监听readystate改变的事件

ajax.onreadystatechange=function(){

if(ajax,readyState == 4 && ajax.status == 200){

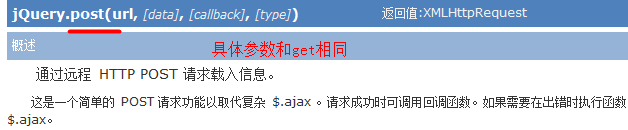
处理正确的响应操作

}

}

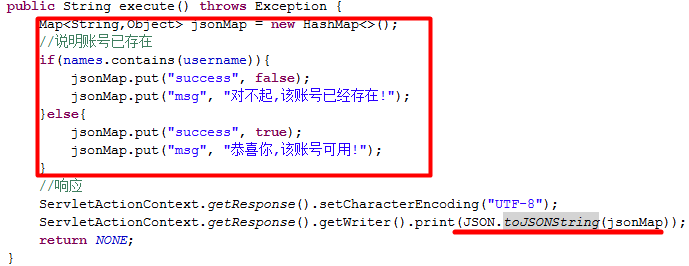
3:发送Ajax请求:

send(***[data]***);



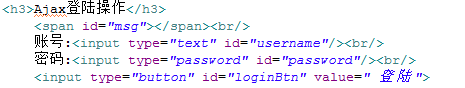
**使用jQuery发送Ajax的GET请求,并传递参数**.

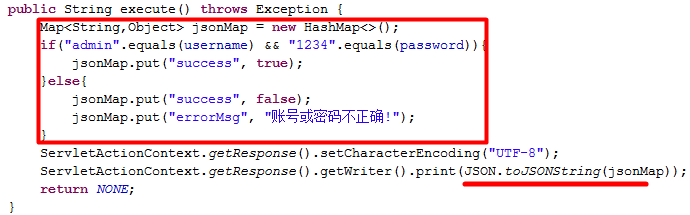
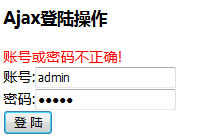
**案例:检查注册账号是否已经存在**.



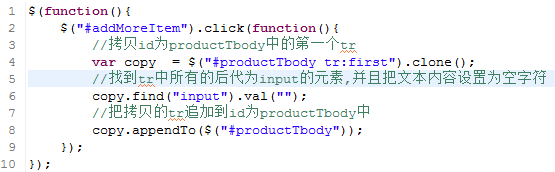
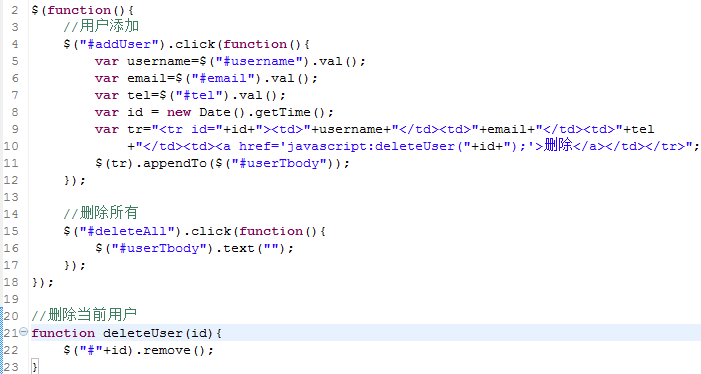
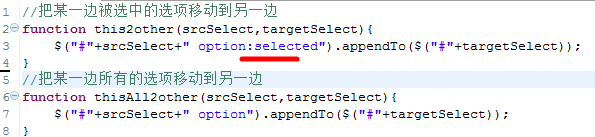
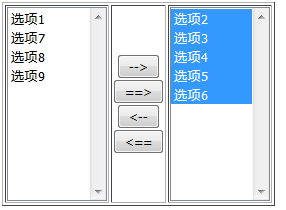
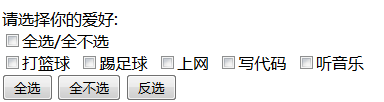
**使用jQuery的Ajax发生POST请求,并传递参数**.

## 案例:使用Ajax完成登陆操作.





## jQuery操作练习



## 小结

