

-

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

# HTML

## 入门

### 1．html简介

html是什么

html是用来描述网页的一种语言

html值得是超文本标记语言，不是一种编程语言，而是一种标记语言，标记语言是一种标记标签。使用标签来描述网页。超文本：普通文本不能实现的，超文本可以实现，能实现普通文本不能实现的功能。标记：就是标签，不同标签能实现不同的功能。

### 2．Html书写规范

需求：写一段文字 将其中的部分文字变成红色，字号变大。

<html>

<head></head>

<body>

床前明月光，疑似地上霜。 <font color="#ff8c00" size="4">举头望明月低头思故乡</font>

<br/>大家很高兴

</body>

</html>

### 3．html结构

<html>

<head>

包含资讯信息：整个页面的属性，指导浏览器解析的标签，引入外部文件的标签。

</head>

<body>

我们需要展示的信息

</body>

</html>

1)html标签是一尖括号包裹关键字成对出现的，有开始标签，有结束标签。 支持正确的嵌套。

2)大部分标签有属性，格式：属性=“属性值” 多个属性值之间用空格键隔开。

3)空标签 功能比较单一，<br><br/>==<br/>

4)html标签不区分大小写。

<html>

<head>

包含资讯信息：整个页面的属性，指导浏览器解析的标签，引入外部文件的标签。

</head>

<body>

我们需要展示的信息

</body>

</html>

2.html标签是一尖括号包裹关键字成对出现的，有开始标签，有结束标签。 支持正确的嵌套。

3.大部分标签有属性，格式：属性=“属性值” 多个属性值之间用空格键隔开。

4.空标签 功能比较单一，<br><br/>==<br/>

5.html标签不区分大小写。

## 基本标签

### 1.文件标签（结构标签）

<html>：根标签

<head>

<title>:页面标题

<body>：内容

属性：

Text：文本的颜色

Bgcolor：背景色

Background：背景图片

颜色的写法: 1)单词： red green black

2)reg三原色:red(0,0,0) 0-255

3)#000000 #ffffff #123132

### 2.排版标签

<!--

1)注释规范

2)换行标签<br/>

3)段落标签 <p>文本文字</p> 断语段之间有空行

属性：align：对齐 left center right

4)水平线 <hr/>

属性：width:长度

size:粗度

color：颜色

align：对齐

尺寸的写法：

像素:10px

百分比:占据标签的百分比:50%

区别：相对和绝对

-->

### 3.块标签

div：行级块标签

span：行块标签

作用：div+css布局

span：进行友好提示

### 4.文字标签

基本文字标签：<front>

属性：

color：颜色

size：大小

最大值7

最小值1

默认值3

face：字体的类型

标题标签：

<h1>-<h6>

标题标签是依次减小的，默认字体的加粗，内置字号，默认占据一行

### 5.清单标签

unorderly：<ul>

attribute:type:disc square circle

<li>:List items

orderly：<ol>

attribute:type:1、A、I、a、i

start:开始的位置

<li>：列表项

作用:可以实现横向和纵向菜单

### 6.图形标签

<img>

src:address of image

width:宽度

height:高度

border:边框

align:对齐方式 代表图片与相邻的文本的相对位置 top middle bottom

alt:图片的文字说明

### 1. 链接标签

<table>

attribute:

border:table border

width:表格的宽度

align：表格的对齐方式

bgcolor:背景颜色

<tr> 代表行

<td> 代表单元格

attribute:

colspan:列合并

rowspan:行合并

<caption> 表格的标题

affect:

1.简单的实现一个表格样式

2.进行布局

<thead>

<tbody>

<tfoot>

effect:分块加载，增强用户体验。

### 2. 表格标签

<table>

attribute:

border:table border

width:表格的宽度

align：表格的对齐方式

bgcolor:背景颜色

<tr> 代表行

<th>加粗居中

<td> 代表单元格

attribute:

colspan:列合并

rowspan:行合并

<caption> 表格的标题

affect:

1.简单的实现一个表格样式

2.进行布局

<thead>

<tbody>

<tfoot>

effect:分块加载，增强用户体验。

## 表单标签（重点）

### 1、form标签

<1.form标签

<form></form>

attribute:

name:表单名称

action:提交的路径地址

method:提交方式get post

\*\*\*\*get和post的difference

1)get提交将数据家在地址栏的后面 格式是:?name=values&name=value 而post提交将数据封装在请求体中

2)get提交相对不安全， post提交相对安全

3)get提交有大小限制，根据浏览器不同而不同， post不限制大小

### 2.input标签

attribute:

type:根据type值的不同会显示不同功能的表单项

text:普通的文本输入框

password:密码输入框，feature：显示掩码

radio:单选按钮 attention:组的概念：如果想让一组单选按钮互斥，必须让name属于同一组

checked:代表默认被选中

checkbox:复选框 attention:组的概念：如果想让一组单选按钮互斥，必须让name属于同一组

checked:代表默认被选中

file:上传文件

button:普通的按钮 没有任何内置功能

reset:重置按钮 点击会将表单清空

image:图片按钮 点击会按照action地址进行提交

属性:

src:图片的地址

alt:图片的文字信息

hidden:隐藏标签: effect:服务端需要但是不需要用户看到

### 3.select标签

name：表单项的名称

option:代表一个选择项

attribute:

value：

selected:默认被选中的项

### 4.textarea标签

attribute:

col:列数

rows:行数

attention：默认的文本值在标签体

# Css

## 不可见

type="hidden"

<input name="id" value="${msUser.id }" type="hidden"/>

<div style="display:none"></style>，完全隐藏，不占空间

<div style="visibility:hidden"></style> 不可见，但是还占空间

## 圆滑边缘

style="border-radius:20px"

## 置顶

<div style='position:fixed; [z-index](https://www.baidu.com/s?wd=z-index&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao):999; top:0;'></div>相对浏览器  
  
<div style='position:absolute [z-index](https://www.baidu.com/s?wd=z-index&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao):999; top:0;'></div>相对父元素  
  
<div style='position:relative; [z-index](https://www.baidu.com/s?wd=z-index&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao):999; top:0;'></div>相对自己

# Bootstrap

今日任务

使用JQuery发送请求局部刷新页面

使用BootStrap制作一个响应式的页面

使用BootStrap制作网站的首页

教学目标

掌握什么是响应式及响应式的原理

掌握BootStrap的栅格系统

了解BootStrap的其他组件及JS控件

上次课内容:

什么JQ : write less do more 写更少的代码,做更多的事情 javascript函数库

基本选择器:

​ ID选择器: #ID名称

​ 类选择器: .类名

​ 元素选择器: 元素的名称

​ 通配符选择器: \* 找出页面上所有元素

​ 选择器分组: 选择器1,选择器2 [选择器1 , 选择器2]

层级选择器:

​ 后代选择器: 选择器1 选择器2 找出来的选择器1 下面的所有选择器2 子孙

​ 子元素选择器: 选择器1 > 选择器2 找出来的是所有的子节点 儿子

​ 相邻兄弟选择器: 选择器1+选择器2 找出来的紧挨着自己的弟弟

​ 兄弟选择器: 选择器1~选择器2 找出所有的弟弟

​ (找出所有兄弟: $("div").siblings() )

属性选择器:

选择器 div

选择器[title]

选择器[title='test']

选择器[title='test'][style]

基本的过滤器: 选择器:过滤器 $("div:first")

​ :first : 找出第一个元素

​ :last 找出最后一个元素

​ :even 找出偶数索引

​ :odd 找出奇数

​ :gt(index) greater-than大于

​ :lt(index) 小于

​ :eq(index) 等于

表单选择器:

​ :input 找出所有的输入项, textarea select button

​ :password

​ :text

​ :radio

表单对象属性的过滤器

​ :selected

​ :checked

常用函数:

​ 属性prop() properties

​ 如果传入一个参数 就是获取

​ prop("src","../img/1.jpg");

​ 设置图片路径

​ attr : 操作一些自定义的属性

​ prop: 通常是用来操作元素固有属性的 ,建议大家使用prop来操作属性

​ css() ; 修改css样式

​ addClass() : 添加一个class样式

​ removeClass() : 移除

​

​ blur : 绑定失去焦点

​ focus: 绑定获得焦点事件

​ click:

​ dblclick

​ change

​

​ append : 给自己添加儿子

​ appendTo : 把自己添加到别人家

​ prepend : 在自己子节点最前面添加子节点

​ after : 在自己后面添加一个兄弟

​ before: 在自己前面添加一个兄弟

​

​ $("数组对象").each(function(index,data))

​ $.each(arr,function(index,data))

​

## 表单校验案例

技术分析

trigger : 触发事件,但是会执行类似浏览将光标移到输入框内的这种浏览器默认行为

triggerHandler : 仅仅只会触发事件所对应的函数

is()

步骤分析

首先给必填项,添加尾部添加一个小红点

获取用户输入的信息,做相应的校验

事件: 获得焦点, 失去焦点, 按键抬起

表单提交的事件

代码实现

使用JQuery发送请求局部刷新页面

​ 数据交换格式:

​ json

​ xml

​什么是BootStrap

​

BootStrap有什么作用

复制粘贴, 能够提高开发人员的工作效率

什么是响应式页面

适应不同的分辨率显示不同样式,提高用户的体验

​

BootStrap的中文网

http://www.bootcss.com

下载BootStrap

BootStrap结构

全局CSS

bootStrap中已经定义好了一套CSS的样式表

组件

BootStrap定义的一套按钮,导航条等组件

JS插件

BootStrap定义了一套JS的插件,这些插件已经默认实现了很多种效果

BootStrap的入门开发

引入相关的头文件

<!-- 最新版本的 Bootstrap 核心 CSS 文件 -->

<link rel="stylesheet" href="../css/bootstrap.css" />

<!--需要引入JQuery-->

<script type="text/javascript" src="../js/jquery-1.11.0.js" ></script>

<!-- 最新的 Bootstrap 核心 JavaScript 文件 -->

<script type="text/javascript" src="../js/bootstrap.js" ></script>

<meta name="viewport" content="width=device-width, initial-scale=1">

BootStrap的布局容器

.container 类用于固定宽度并支持响应式布局的容器。

<div class="container">

...

</div>

.container-fluid 类用于 100% 宽度，占据全部视口（viewport）的容器。

<div class="container-fluid">

...

</div>

校验表单扩展:

trigger : 触发浏览器默认行为

triggerHandler : 不会触发

is : 判断

find : 查找

老黄历:

什么json: 轻量级的数据交换格式

json对象: {"username":"zhangsan"}

json数组: [ {"username":"zhangsan"}, {"username":"zhangsan"}, {"username":"zhangsan"}]

ajax异步请求:

​ 同步和异步

row

Bootstrap 栅格系统的工作原理：

“行（row）”必须包含在 .container （固定宽度）或 .container-fluid （100% 宽度）中，以便为其赋予合适的排列（aligment）和内补（padding）。

通过“行（row）”在水平方向创建一组“列（column）”。

你的内容应当放置于“列（column）”内，并且，只有“列（column）”可以作为行（row）”的直接子元素。

类似 .row 和 .col-xs-4 这种预定义的类，可以用来快速创建栅格布局。Bootstrap 源码中定义的 mixin 也可以用来创建语义化的布局。

通过为“列（column）”设置 padding 属性，从而创建列与列之间的间隔（gutter）。通过为 .row 元素设置负值 margin 从而抵消掉为 .container 元素设置的 padding，也就间接为“行（row）”所包含的“列（column）”抵消掉了padding

​

### BootStrap的栅格系统

响应式设计: 这种设计依赖于CSS3中的媒体查询

栅格样式:

设备分辨率大于1200 使用lg样式

设备分辨率大于992 < 1200 使用md样式

设备分辨率大于768 < 992 使用sm样式

设备分辨率小于768使用xs样式

BootStrap的全局CSS

定义了一套CSS

对页面中的元素进行定义

列表元素,表单,按钮,图片...

步骤分析

代码实现

使用BootStrap布局网站首页

### 需求分析

请使用BootStrap对我们的首页进行优化

技术分析

步骤分析

新建一个HTML页面.引入bootStrap相关的js和CSS

定义一个整体的div, 将整体的div分成8个部分

完成没部分的内容显示

代码实现

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<!--

准备工作:

<meta name='viewport'>

1.导入bootstrap css文件

2.导入JQuery

3.bootstrap.js

4.写一个div class = container 支持响应式的布局容器

-->

<link rel="stylesheet" href="../css/bootstrap.min.css">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!--

jQuery文件。务必在bootstrap.min.js 之前引入

-->

<script src="../js/jquery-1.11.0.js"></script>

<!-- 最新的 Bootstrap 核心 JavaScript 文件 -->

<script src="../js/bootstrap.min.js"></script>

</head>

<body>

<div class="container">

<div class="row">

<div class="col-md-4">

<img src="../img/logo2.png" />

</div>

<div class="col-md-4 hidden-xs">

<img src="../img/header.png" />

</div>

<div class="col-md-4">

<a href="#">登录</a>

<a href="#">注册</a>

<a href="#">购物车</a>

</div>

</div>

<!--菜单-->

<div class="row">

<div class="col-md-12">

<nav class="navbar navbar-inverse" role="navigation">

<div class="container-fluid">

<!-- Brand and toggle get grouped for better mobile display -->

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="#">首页</a>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

<ul class="nav navbar-nav">

<li class="active">

<a href="#">手机数码</a>

</li>

<li>

<a href="#">鞋靴箱包</a>

</li>

<li>

<a href="#">电脑办公</a>

</li>

<li class="dropdown">

<a href="#" class="dropdown-toggle" data-toggle="dropdown">所有分类 <span class="caret"></span></a>

<ul class="dropdown-menu" role="menu">

<li>

<a href="#">手机数码</a>

</li>

<li>

<a href="#">鞋靴箱包</a>

</li>

<li>

<a href="#">电脑办公</a>

</li>

<li class="divider"></li>

<li>

<a href="#">Separated link</a>

</li>

<li class="divider"></li>

<li>

<a href="#">One more separated link</a>

</li>

</ul>

</li>

</ul>

<form class="navbar-form navbar-right" role="search">

<div class="form-group">

<input type="text" class="form-control" placeholder="请输入要搜索的商品">

</div>

<button type="submit" class="btn btn-default">搜索</button>

</form>

</div>

<!-- /.navbar-collapse -->

</div>

<!-- /.container-fluid -->

</nav>

</div>

</div>

<div>

<div id="carousel-example-generic" class="carousel slide" data-ride="carousel">

<!-- Indicators -->

<ol class="carousel-indicators">

<li data-target="#carousel-example-generic" data-slide-to="0" class="active"></li>

<li data-target="#carousel-example-generic" data-slide-to="1"></li>

<li data-target="#carousel-example-generic" data-slide-to="2"></li>

</ol>

<!-- Wrapper for slides -->

<div class="carousel-inner" role="listbox">

<div class="item active">

<img src="../img/1.jpg" alt="...">

<div class="carousel-caption">

...

</div>

</div>

<div class="item">

<img src="../img/2.jpg" alt="...">

<div class="carousel-caption">

...

</div>

</div>

<div class="item">

<img src="../img/3.jpg" alt="...">

<div class="carousel-caption">

...

</div>

</div>

</div>

<!-- Controls -->

<a class="left carousel-control" href="#carousel-example-generic" role="button" data-slide="prev">

<span class="glyphicon glyphicon-chevron-left"></span>

<span class="sr-only">Previous</span>

</a>

<a class="right carousel-control" href="#carousel-example-generic" role="button" data-slide="next">

<span class="glyphicon glyphicon-chevron-right"></span>

<span class="sr-only">Next</span>

</a>

</div>

</div>

<!--最新商品这里-->

<div class="row">

<div class="col-md-12">

<h3>最新商品<img src="../images/title2.jpg"/></h3>

</div>

</div>

<!--商品部分 -->

<div class="row">

<!--左边div-->

<div class="col-md-2 hidden-sm hidden-xs">

<img src="../products/hao/big01.jpg" width="100%" height="100%" />

</div>

<!--右边div-->

<div class="col-md-10">

<!--上面部分-->

<div class="row">

<!--中等广告图-->

<div class="col-md-6">

<img src="../products/hao/middle01.jpg" width="100%" />

</div>

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

</div>

<!--下面部分-->

<div class="row">

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

</div>

</div>

</div>

<div class="row">

<div class="col-md-12">

<img src="../products/hao/ad.jpg" width="100%" />

</div>

</div>

<!--最新商品这里-->

<div class="row">

<div class="col-md-12">

<h3>最新商品<img src="../images/title2.jpg"/></h3>

</div>

</div>

<!--商品部分 -->

<div class="row">

<!--左边div-->

<div class="col-md-2 hidden-sm hidden-xs">

<img src="../products/hao/big01.jpg" width="100%" height="100%" />

</div>

<!--右边div-->

<div class="col-md-10">

<!--上面部分-->

<div class="row">

<!--中等广告图-->

<div class="col-md-6">

<img src="../products/hao/middle01.jpg" width="100%" />

</div>

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

</div>

<!--下面部分-->

<div class="row">

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

<!--商品项-->

<div class="col-md-2 col-xs-4" style="text-align: center;">

<img src="../products/hao/small01.jpg" />

<p>豆浆机</p>

<p>$998</p>

</div>

</div>

</div>

</div>

<!--页脚广告-->

<div>

<img src="../image/footer.jpg" width="100%" />

</div>

<!--网站声明-->

<div style="text-align: center;">

<a href="http://www.itheima.com">关于我们</a>

<a href="http://www.itheima.com">联系我们</a>

<a href="http://www.itheima.com">招贤纳士</a>

<a href="http://www.itheima.com">法律声明</a>

<a href="http://www.itheima.com">友情链接</a>

<a href="http://www.itheima.com">支付方式</a>

<a href="http://www.itheima.com">配送方式</a>

<a href="http://www.itheima.com">服务声明</a>

<a href="http://www.itheima.com">广告声明</a>

<br />

Copyright © 2005-2016 传智商城 版权所有

</div>

</div>

</body>

</html>

五天前端内容总结

JQ方式校验表单(要求做出来)

json : (了解)

json对象 {}

json数组 [{},{}]

$.get(url,function(data){}) (了解)

bootstrap: Bootstrap 是最受欢迎的 HTML、CSS 和 JS 框架，用于开发响应式布局、移动设备优先的 WEB 项目。

全局CSS样式: css样式

栅格系统:

将屏幕划分成12个格子,12列

class='row' 当前是行

行里面放的是列 col-屏幕分辨率-数字 (每一种分辨率后的数字总和必须是等于12,如果超过12,另起一行)

col-lg-数字: 在超宽屏幕上使用

col-md-数字: 在中等屏幕上,PC电脑

col-sm-数字: 在平板电脑上

col-xs-数字: 在手机上

组件: 导航条 , 进度条, 字体

javascript插件 : 轮播图

复制粘贴

什么是响应式: 会根据不同的分辨率去显示不同页面结构,提高用户体验

HTML: 超文本标记语言: 设计网页,决定了网页的结构

CSS: 层叠样式表 ,主要是用来美化页面, 将美化和HTML代码进行分离,提高代码复用性

javascript: 脚本语言,由浏览器解释执行, 弱类型语言(var i), 提供用户交互

jquery: javascript函数库,进一步的封装

选择器:

ID选择器

类选择器

元素选择器

通配符选择器

选择器分组

层级选择器

后代选择器

子元素选择器

相邻兄弟选择器

兄弟选择器 : 找出所有的弟弟

属性选择器:

选择器[属性名称='属性的值']

表单选择器

:input

:text

:password

body > div > div:nth-child(7) > div:nth-child(3) > div:nth-child(8)

基本的过滤器

:first

:last

:even

:odd

:gt

:lt

:eq

表单对象属性

## 警告框

$(".close").click(function(){

$("#test").alert();

}

$('#identifier').alert('close');

手动

$('.alert-danger').removeClass('hide').addClass('in')

<style>

#myAlert {

transition: margin-top .15s linear, opacity .15s linear;

}

#alertBox {

position: absolute;

left: 50%;

margin-left: -400px;

height: 100px;

width: 800px;

}

</style>

<div class="container">

<div id="alertBox">

<div class="alert alert-danger" id="myAlert" style="opacity: 1;">

<a href="#" class="close" data-dismiss="alert">

×

</a>

<strong>警告！</strong>您的网络连接有问题。

</div>

</div>

<button class="btn btn-default" onclick="showFun();">show</button>

<button class="btn btn-default" onclick="closeFun();">close</button>

</div>

<script>

// 方法

function showFun() {

$("#myAlert").css("margin-top", "0px");

$("#myAlert").css("opacity", 1);

}

function closeFun() {

$("#myAlert").css("margin-top", "10px");

$("#myAlert").css("opacity", 0);

//$("#myAlert").alert('close'); //关闭所有的警告框,整个消息框被移除

}

// 事件

$("#myAlert").bind('closed.bs.alert', function() {

console.log("警告消息框被关闭。");

});

</script>

## 模态框（Modal）插件

<https://www.runoob.com/bootstrap/bootstrap-modal-plugin.html>

admlte

https://adminlte.io/docs/2.4/js-box-widget

调用带有 id="identifier" 的模态框：

$('#identifier').modal(options)

### 例子

<h2>创建模态框（Modal）</h2>

<!-- 按钮触发模态框 -->

<button class="btn btn-primary btn-lg" data-toggle="modal" data-target="#myModal">开始演示模态框</button>

<!-- 模态框（Modal） -->

<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>

<h4 class="modal-title" id="myModalLabel">模态框（Modal）标题</h4>

</div>

<div class="modal-body">在这里添加一些文本</div>

<div class="modal-footer">

<button type="button" class="btn btn-default" data-dismiss="modal">关闭</button>

<button type="button" class="btn btn-primary">提交更改</button>

</div>

</div><!-- /.modal-content -->

</div><!-- /.modal -->

</div>

# JS

## 我的

### 常见问题

#### [使用onclick跳转到其他页面。使用button跳转到指定url](https://www.cnblogs.com/wujixing/p/5856087.html)

onclick="javascript:window.location.href='aa.htm'"

onclick="location='URL'"  
onclick="window.location.href='?id=11'"

<button type="button" class="btn btn-block btn-default" onclick="location='toApplyMsProduct'">申请秒杀商品</button>

## 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

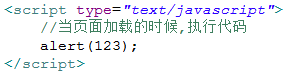
web游戏

服务器脚本开发

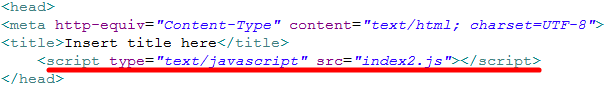
### js代码存在的位置

**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.

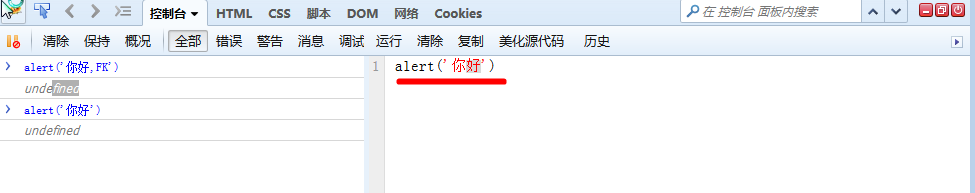


**方式2**:单独使用一个文件来编写javascript代码(**js文件**),在需要使用的页面中引入该文件.



**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";

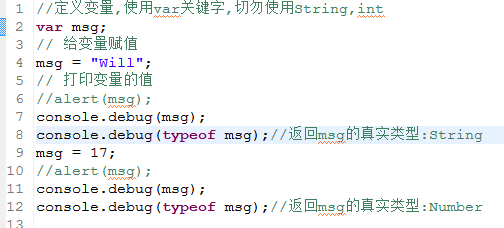
对已赋值的变量赋予一个其他类型的数据

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

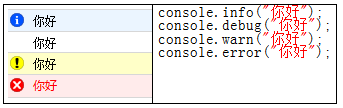


#### 变量的打印:

**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**。

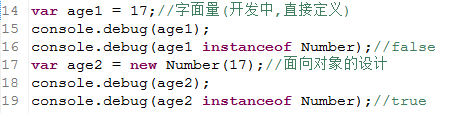
String字符串(注意js中没有char类型，所有'a'也是一个字符串)

“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:

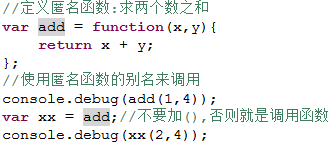
如果在函数中使用了return,那么函数的返回结果就是 return 后的**值**.

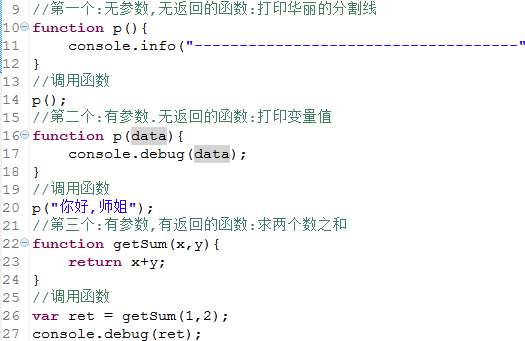
如果在函数中没有使用return,说明该函数没有结果,不用接受,若接受,结果为:undefined.

#### 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.

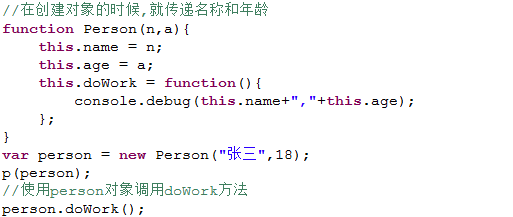
**js创建一个类,只需要定义类的构造函数(方法)**:



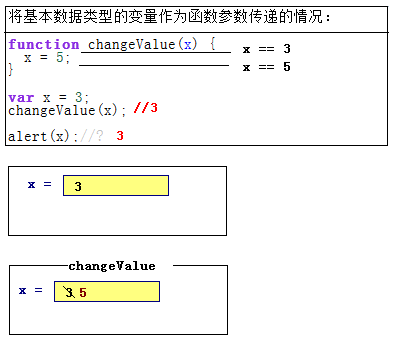
**在JS中this表示什么**? this就表示当前对象

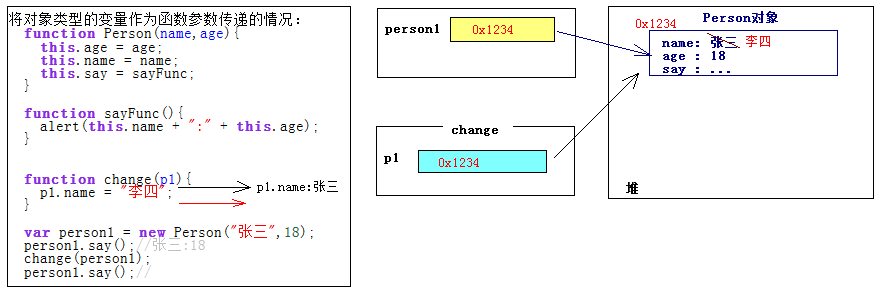
1):this在构造器(类)中,此时this就表示当前被创建的对象.

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

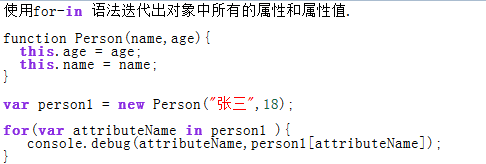
**for in循环好比是Java中的foreach循环**:

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

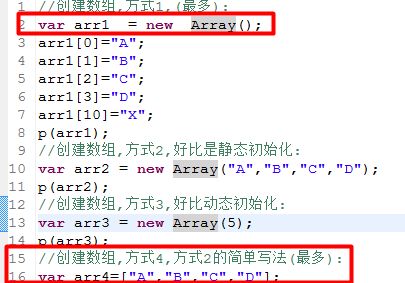


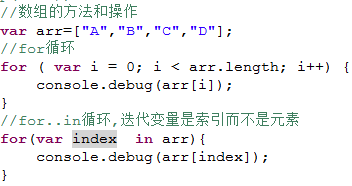
### 数组

#### 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；

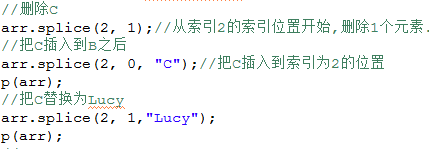
shift－移出第一个元素；

splice－截取数组;

sort－排序数组;

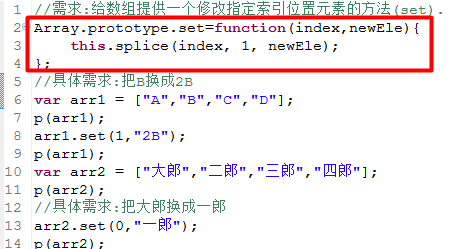
unshift－在前面追加元素；

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



### prototype

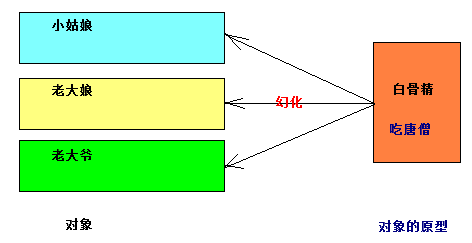
**需求:给数组提供一个修改指定索引位置元素的方法(set)**.



**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.**

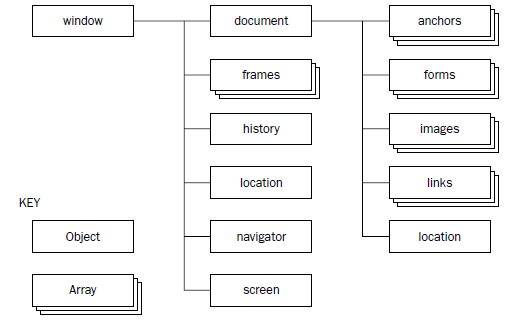
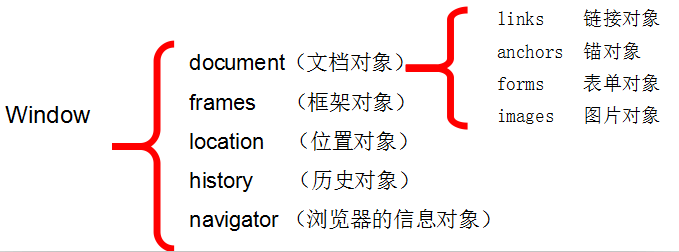


小结



### 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

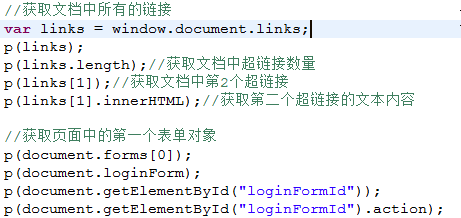
每个载入浏览器的 HTML 文档都会成为 Document 对象。

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

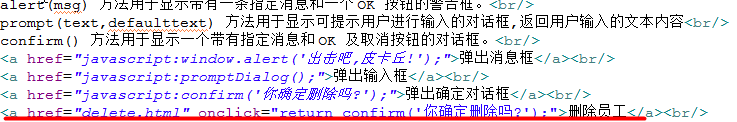
//alert("你好1");

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

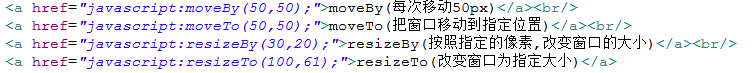
**IE中有效**

// 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;

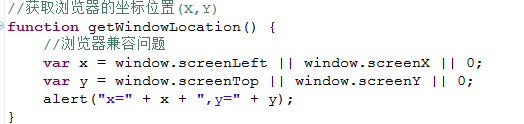
var y = window.screenY || window.screenTop || 0;

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:

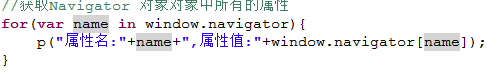
var isWin = (navigator.platform == “Win32”) || (navigator.platform == “Windows”);

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

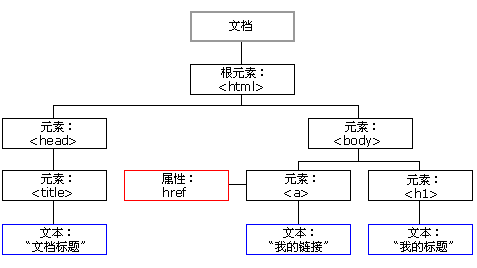
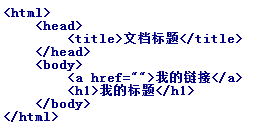
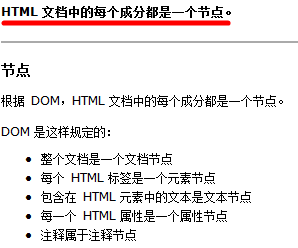
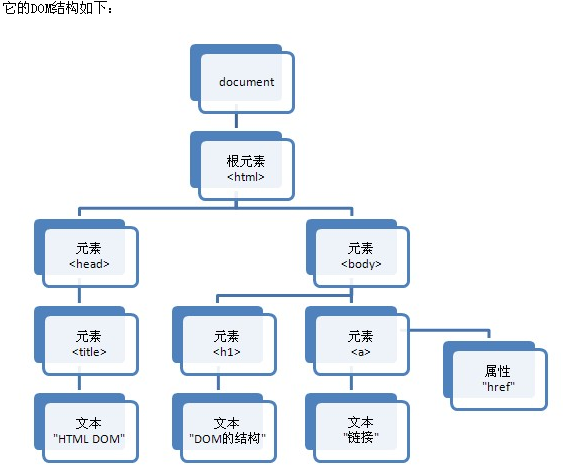
M：模型

DOM 是针对xml(html)的基于树的API。

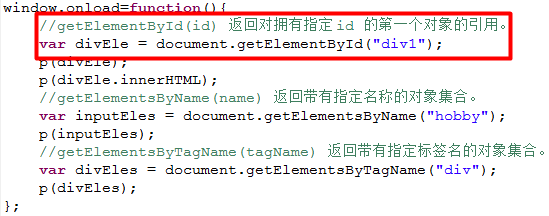
DOM树:节点（node）的层次。

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.

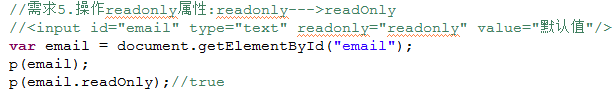
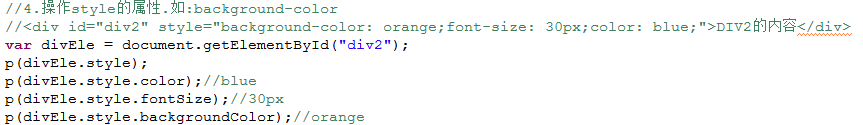
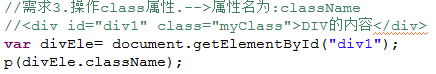
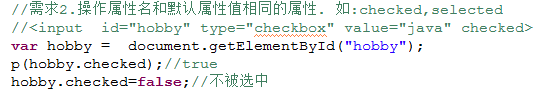
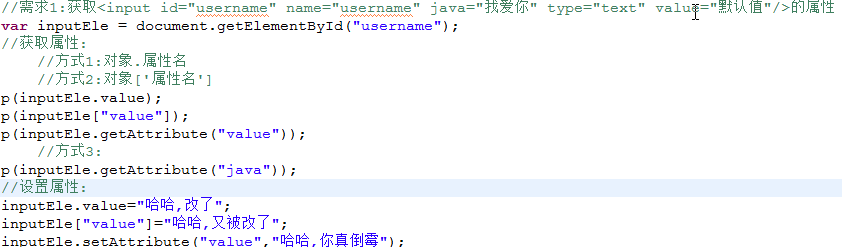
4.操作style的属性.如:background-color

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

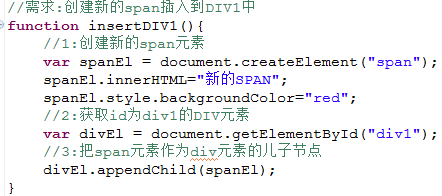
**插入/追加/删除/替换节点**:

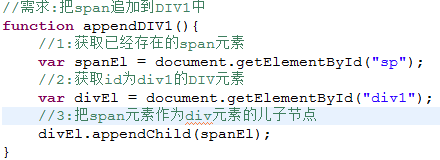
1.给父节点追加现有/新子节点

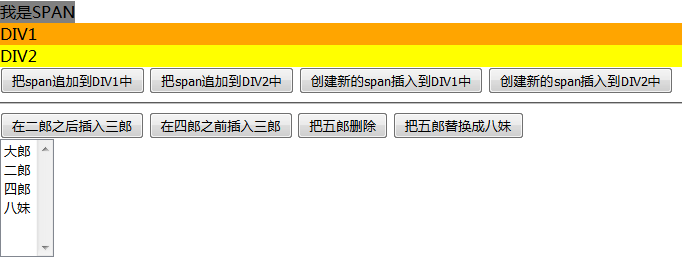
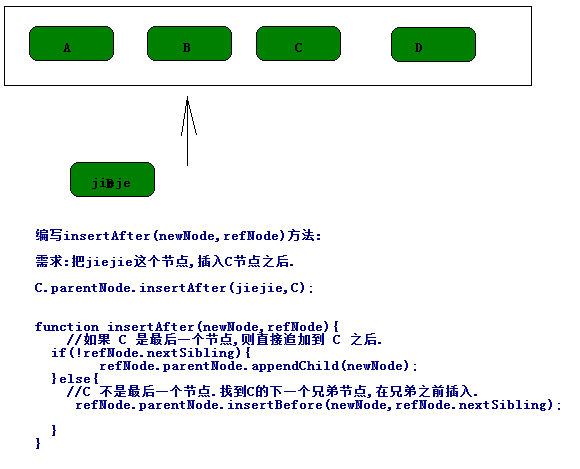
2.在指定节点之前/之后插入新节点

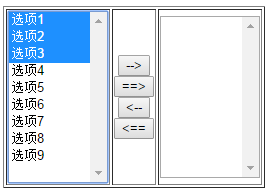
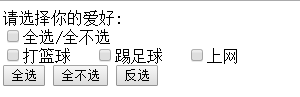
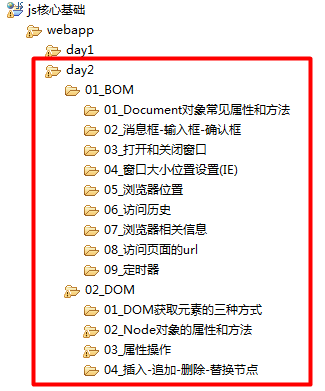
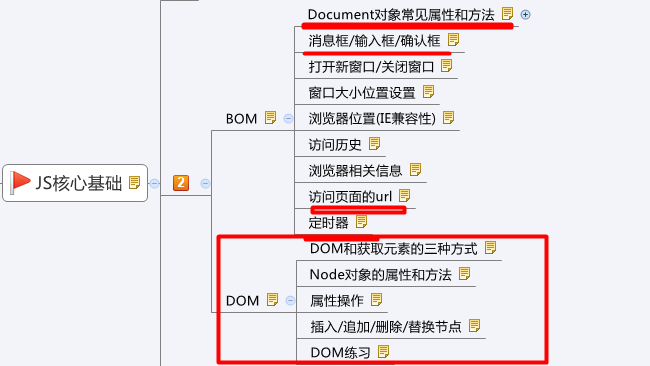
3.删除节点/替换节点









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

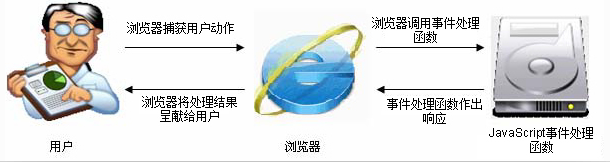
**JavaScript处理事件的基本机制**：

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>

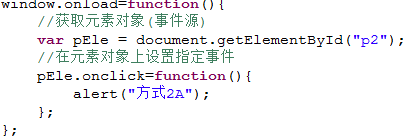


方式1:很简单也很直观,但是同一种的事件只能绑定一次.

**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

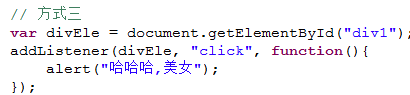
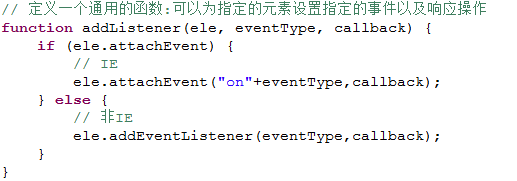
多次添加监听后,触发顺序: 先添加,先执行

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

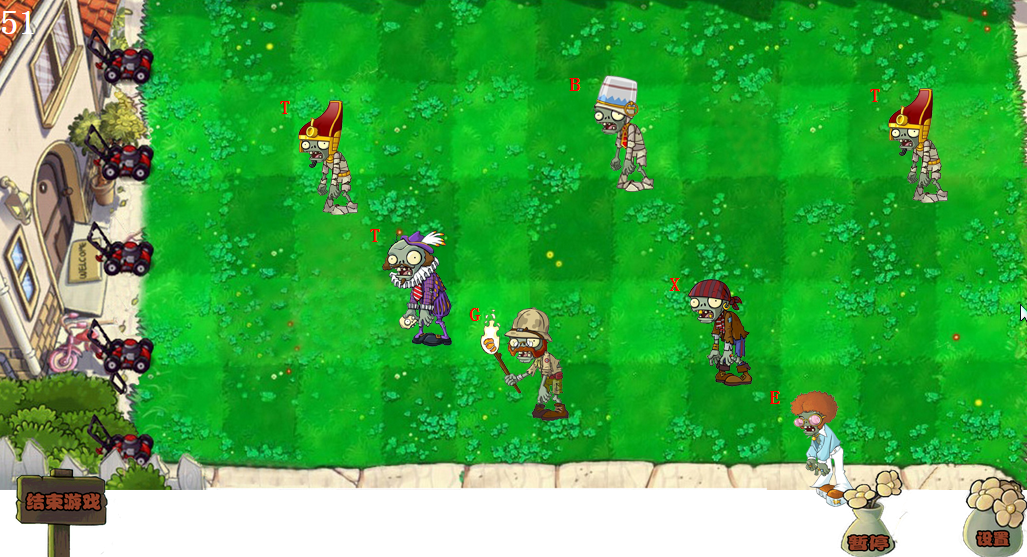
<label style="position:absolute;top:30px;left:10px;background-color:red;width:20px;text-align:center;font-weight:bold;">I</label>

2.1.2> 把<lable>元素添加到body元素中

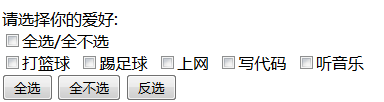
2.2) 让页面中的字符周期性向下移动

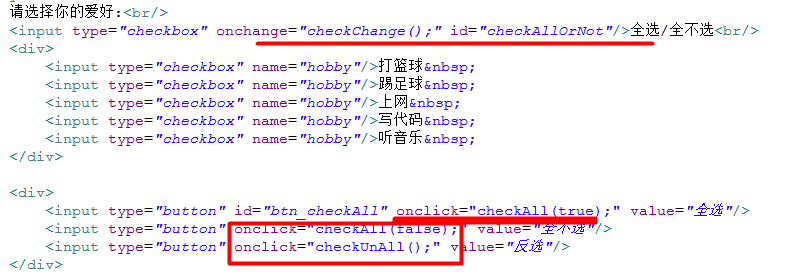
2.3) 对键盘输入内容,监听,对比输入内容,和游戏界面字母内容,一致则消除

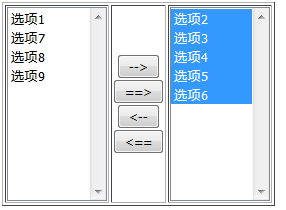


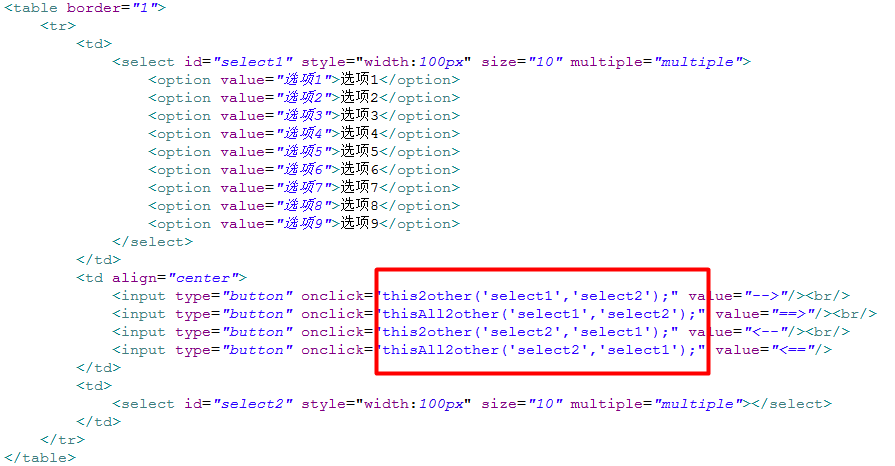


### 练习题







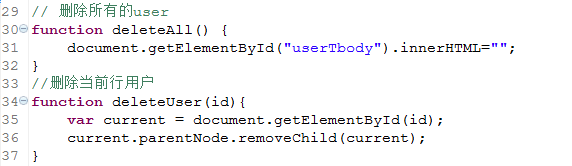




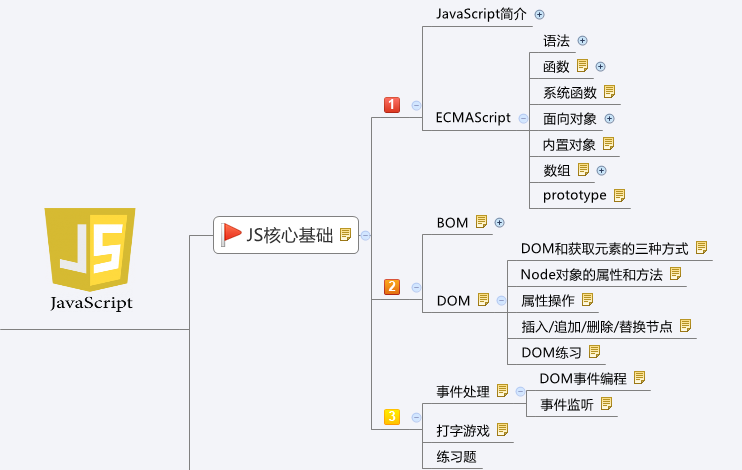








小结



## 常用指令

刷新

window.location.reload();

## 表单验证

function validateForm()

{

var x=document.forms["myForm"]["fname"].value;

if (x==null || x=="")

{

alert("姓必须填写");

return false;

}

}

<form name="myForm" action="demo-form.php" onsubmit="return validateForm()" method="post">

姓: <input type="text" name="fname">

<input type="submit" value="提交">

</form>

**5.1.5 表单验证**

以一个简单的用户注册为例。首先新建一个表单， HTML 代码如下：

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="content-type" content="text/html; charset=utf-8">

<title>jQueryStudy</title>

<style>

.int{ height: 30px; text-align: left; width: 600px; }

label{ width: 200px; margin-left: 20px; }

.high{ color: red; }

.msg{ font-size: 13px; }

.onError{ color: red; }

.onSuccess{ color: green; }

</style>

<script src="http://code.jquery.com/jquery-latest.js"></script>

</head>

<body>

<form method="post" action="#">

<div class="int">

<label for="name">姓名：</label>

<input type="text" id="name" class="required" />

</div>

<div class="int">

<label for="email">邮箱：</label>

<input type="text" id="email" class="required" />

</div>

<div class="int">

<label for="address">住址：</label>

<input type="text" id="address" />

</div>

<div class="int">

<input type="submit" value="提交" id="send" style="margin-left: 70px;" />

<input type="reset" value="重置" id="res" />

</div>

</form>

</body>

</html>

       显示效果如图 1 所示：



                             图 1 表单初始化

     在表单内 class 属性为 “required” 的文本框是必须填写的，因此需要将它与其他的非必须填写的表单元素加以区别，即在文本框后面追加一个红色的星号标识。可以使用 append() 方法来完成，代码如下：

　　 //为表单的必填文本框添加提示信息（选择form中的所有后代input元素）

$("form :input.required").each(function () {

//通过jquery api：$("HTML字符串") 创建jquery对象

var $required = $("<strong class='high'>\*</strong>");

//添加到this对象的父级对象下

$(this).parent().append($required);

});

       显示效果如图 2 所示：



                           图 2 在必填元素后添加小红星标记

      用户在“姓名”文本框中填写完信息后，将光标的焦点从“姓名”文本框中移出时，需要即时判断姓名是否符合验证规则。当光标的焦点从“邮箱”文本框移出时，也需要即时判断“邮箱”填写是否正确，因此需要给表单元素添加失去焦点事件，即 blur 事件。 jQuery 代码如下：

　　//为表单的元素添加失去焦点事件

$("form :input").blur(function () {

// ...省略代码

});

        给表单里的必填元素添加验证功能的步骤如下：

     （1）判断当前失去焦点的元素是“姓名”还是“邮箱”，然后分别处理。

     （2）如果是“姓名”，判断元素的值的长度是否小于 6 ，如果小于 6 ，则用红色字体提醒用户输入不正确，反之，则用绿色提醒用户输入正确。

     （3）如果是“邮箱”，判断元素的值是否符合邮箱的格式，如果不符合，则用红色字体提醒用户输入不正确，反之，则用绿色字体提醒用户输入正确。

     （4）将提醒信息追加到当前元素的父元素下面。

      【注】：为了使用方便，当 text 文本框中的内容格式错误时，程序应尽量把所有提示信息都显示出来，而不是每次验证一种格式后只显示一个提示信息。 不过有时为了提高响应性，也会每次只显示一个提示信息。如：检查用户名是否重复，需要访问数据库。

如何提示，需要依据具体的业务需求来确定。

       根据以上分析，可以写出如下 jQuery 代码：

　　　　 //为表单的必填文本框添加提示信息（选择form中的所有后代input元素）

$("form :input.required").each(function () {

//通过jquery api：$("HTML字符串") 创建jquery对象

var $required = $("<strong class='high'>\*</strong>");

//添加到this对象的父级对象下

$(this).parent().append($required);

});

//为表单元素添加失去焦点事件

$("form :input").blur(function(){

var $parent = $(this).parent();

//验证姓名

if($(this).is("#name")){

var nameVal = $.trim(this.value); //原生js去空格方式：this.replace(/(^\s\*)|(\s\*$)/g, "")

var regName = /[~#^$@%&!\*()<>:;'"{}【】 ]/;

if(nameVal == "" || nameVal.length < 6 || regName.test(nameVal)){

var errorMsg = " 姓名非空，长度6位以上，不包含特殊字符！";

//class='msg onError' 中间的空格是层叠样式的格式

$parent.append("<span class='msg onError'>" + errorMsg + "</span>");

}

else{

var okMsg=" 输入正确";

$parent.find(".high").remove();

$parent.append("<span class='msg onSuccess'>" + okMsg + "</span>");

}

}

//验证邮箱

if($(this).is("#email")){

var emailVal = $.trim(this.value);

var regEmail = /.+@.+\.[a-zA-Z]{2,4}$/;

if(emailVal== "" || (emailVal != "" && !regEmail.test(emailVal))){

var errorMsg = " 请输入正确的E-Mail住址！";

$parent.append("<span class='msg onError'>" + errorMsg + "</span>");

}

else{

var okMsg=" 输入正确";

$parent.find(".high").remove();

$parent.append("<span class='msg onSuccess'>" + okMsg + "</span>");

}

}

});

       当连续几次输入错误的格式后，会出现如图 3 所示的效果

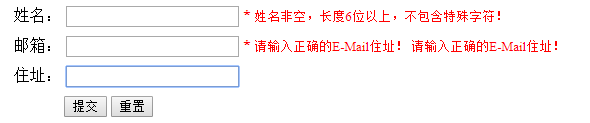


                                    图 3 操作多次的提示效果

      由于每次在元素失去焦点后，都会创建一个新的提醒元素，然后将它追加到文档中，最后就出现了多次的提醒信息。因此，需要在创建提醒元素之前，将当前元素以前的提醒元素都删除。可以使用 remove() 方法来完成，代码如下：

//为表单元素添加失去焦点事件

$("form :input").blur(function(){

var $parent = $(this).parent();

$parent.find(".msg").remove(); //删除以前的提醒元素（find()：查找匹配元素集中元素的所有匹配元素）

// ...省略代码

});

      显示效果如图 4 所示。



                               图 4 正确的效果

      当鼠标在表单元素中多次失去焦点时，都可以提醒用户填写是否正确。但是，如果用户无视错误提醒，执意要单击“提交”按钮时，为了使表单填写准确，在表单提交之前，需要对表单的必须填写元素进行一次整体的验证。可以直接用 trigger() 方法来触发blur事件，从而达到验证效果。如果填写错误，就会以红色提醒用户；如果姓名和邮箱都不符合规则，那么就有两处错误，即有两个 class 为 “onError” 的元素，因此可以根据 class 为 “onError” 元素的长度来判断是否可以提交。如果长度为 0 ，即 true ，说明已经可以提交了；如果长度大于0，即 false ，说明有错误，需要阻止表单提交。阻止表单提交可以直接用 “return false” 语句。

      根据上面的分析，可以在提交事件中写出如下jQuery代码：

//点击重置按钮时，通过trigger()来触发文本框的失去焦点事件

$("#send").click(function(){

//trigger 事件执行完后，浏览器会为submit按钮获得焦点

$("form .required:input").trigger("blur");

var numError = $("form .onError").length;

if(numError){

return false;

}

alert("注册成功！");

});

      显示效果如图 5 所示。



                          图 5 验证通过

      用户也许会提出：为什么每次都要等字段元素失去焦点后，才提醒是否正确？如果输入时就可以提醒，这样就可以更加即时了。

      为了达到用户提出的需求，需要给表单元素绑定 keyup 事件和 focus 事件，keyup 事件能在用户每次松开按键时触发，实现即时提醒；focus 事件能在元素得到焦点的时候触发，也可以实现即时提醒。

      代码如下：

$("form :input").blur(function(){

//失去焦点处理函数

//代码省略...

}).keyup(function(){

//triggerHandler 防止事件执行完后，浏览器自动为标签获得焦点

$(this).triggerHandler("blur");

}).focus(function(){

$(this).triggerHandler("blur");

});

这样当用户将光标定位到文本框上和改变文本框的值时，表单就会即时提醒用户填写是否正确，从而符合了用户的需求。

在前面的章节已经提过，trigger（“blur”）不仅会触发为元素绑定的blur事件，也会触发浏览器默认的 blur 事件，即不能讲光标定位到文本框上。而 triggerHandler（“blur”）只会触发为元素绑定的 blur 事件，而不触发浏览器默认的 blur 事件。

至此，表单验证过程就全部完成。读者可以根据自己的实际需求修改验证规则。

【注】：客户端的验证仅用于提升用户的操作体验，而服务器仍需对用户输入的数据的合法性进行校验。对于禁用了脚本的用户和用户自制的网页提交操作，必须在服务器端验证。

**完整代码：**

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="content-type" content="text/html; charset=utf-8">

<title>jQueryStudy</title>

<style>

.int{ height: 30px; text-align: left; width: 600px; }

label{ width: 200px; margin-left: 20px; }

.high{ color: red; }

.msg{ font-size: 13px; }

.onError{ color: red; }

.onSuccess{ color: green; }

</style>

<script src="http://code.jquery.com/jquery-latest.js"></script>

</head>

<body>

<form method="post" action="#">

<div class="int">

<label for="name">姓名：</label>

<input type="text" id="name" class="required" />

</div>

<div class="int">

<label for="email">邮箱：</label>

<input type="text" id="email" class="required" />

</div>

<div class="int">

<label for="address">住址：</label>

<input type="text" id="address" />

</div>

<div class="int">

<input type="submit" value="提交" id="send" style="margin-left: 70px;" />

<input type="reset" value="重置" id="res" />

</div>

</form>

<script>

//为表单的必填文本框添加提示信息（选择form中的所有后代input元素）

$("form :input.required").each(function () {

//通过jquery api：$("HTML字符串") 创建jquery对象

var $required = $("<strong class='high'>\*</strong>");

//添加到this对象的父级对象下

$(this).parent().append($required);

});

//为表单元素添加失去焦点事件

$("form :input").blur(function(){

var $parent = $(this).parent();

$parent.find(".msg").remove(); //删除以前的提醒元素（find()：查找匹配元素集中元素的所有匹配元素）

//验证姓名

if($(this).is("#name")){

var nameVal = $.trim(this.value); //原生js去空格方式：this.replace(/(^\s\*)|(\s\*$)/g, "")

var regName = /[~#^$@%&!\*()<>:;'"{}【】 ]/;

if(nameVal == "" || nameVal.length < 6 || regName.test(nameVal)){

var errorMsg = " 姓名非空，长度6位以上，不包含特殊字符！";

//class='msg onError' 中间的空格是层叠样式的格式

$parent.append("<span class='msg onError'>" + errorMsg + "</span>");

}

else{

var okMsg=" 输入正确";

$parent.find(".high").remove();

$parent.append("<span class='msg onSuccess'>" + okMsg + "</span>");

}

}

//验证邮箱

if($(this).is("#email")){

var emailVal = $.trim(this.value);

var regEmail = /.+@.+\.[a-zA-Z]{2,4}$/;

if(emailVal== "" || (emailVal != "" && !regEmail.test(emailVal))){

var errorMsg = " 请输入正确的E-Mail住址！";

$parent.append("<span class='msg onError'>" + errorMsg + "</span>");

}

else{

var okMsg=" 输入正确";

$parent.find(".high").remove();

$parent.append("<span class='msg onSuccess'>" + okMsg + "</span>");

}

}

}).keyup(function(){

//triggerHandler 防止事件执行完后，浏览器自动为标签获得焦点

$(this).triggerHandler("blur");

}).focus(function(){

$(this).triggerHandler("blur");

});

//点击重置按钮时，通过trigger()来触发文本框的失去焦点事件

$("#send").click(function(){

//trigger 事件执行完后，浏览器会为submit按钮获得焦点

$("form .required:input").trigger("blur");

var numError = $("form .onError").length;

if(numError){

return false;

}

alert("注册成功！");

});

</script>

</body>

</html>

## 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)**.

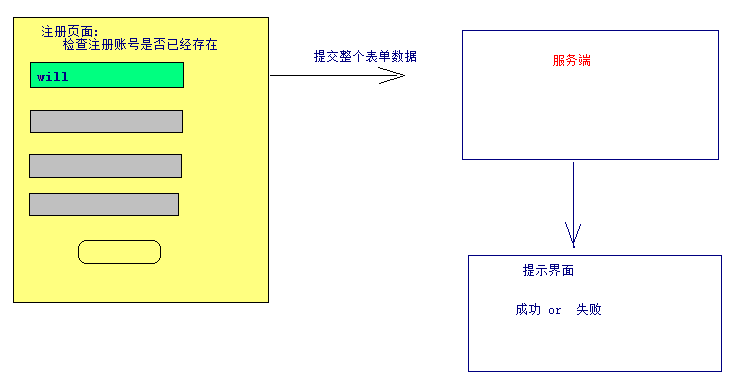
-------------------------------------------------------------------------

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**

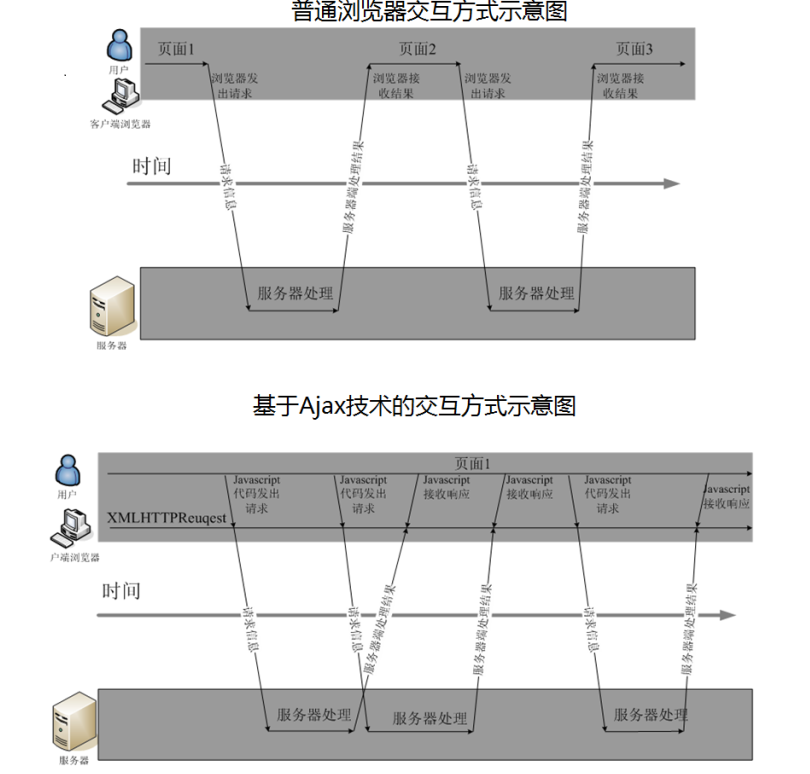
**器处理**->处理完毕返回 这个期间客户端**浏览器不能干任何事(单线程)**.

\* **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**：

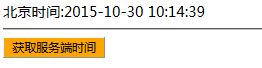
1，AJAX大量使用了Javascript和AJAX引擎，而这个取决于浏览器的支持。IE5.0及以上、Mozilla1.0、NetScape7及以上版本才支持，Mozilla虽然也支持AJAX，但是提供XMLHttpRequest的方式不一样。所以，使用AJAX的程序必须测试针对各个浏览器的兼容性。

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”。

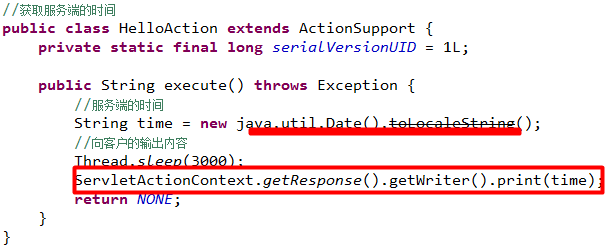
3，对流媒体的支持没有FLASH、Java Applet好。

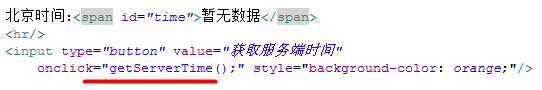
::::**html5 + Ajax(jQuery) 取代 Flash**.

### 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**.

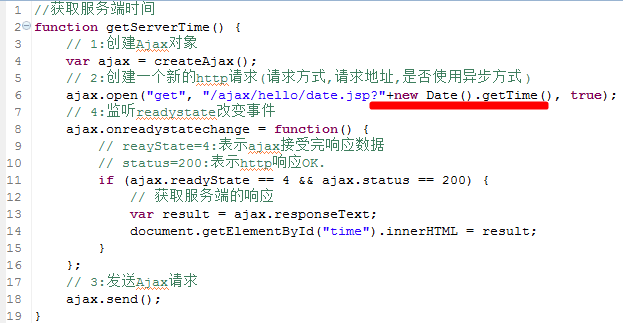
针对于IE6和IE6以下的版本不支持.

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).

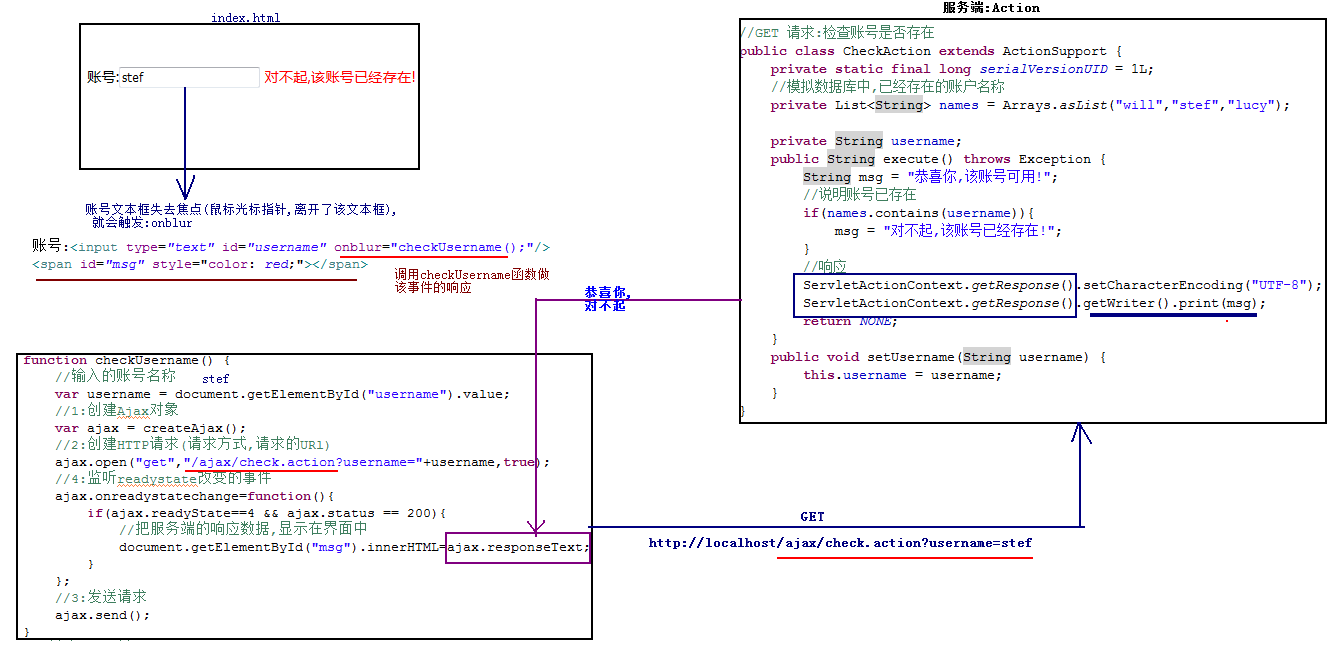
**解决方案:使用时间戳作为资源的参数**.



### 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,

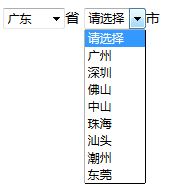
**我假设它有两个下拉列表(其实还可以有更多)**

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**.

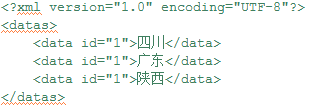
--------------------------------------------------------------------

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.

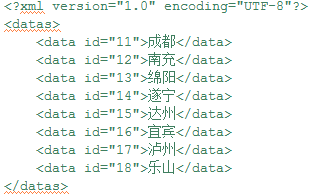


**省份的XML数据格式**:

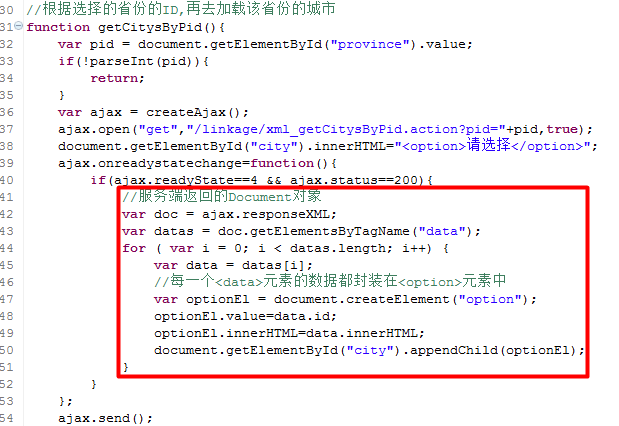




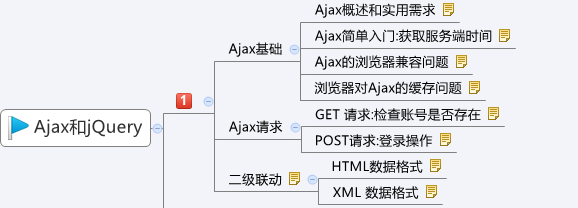
**城市的XML数据格式**:







小结



**熟练完成Ajax的知识点**:

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

自行预习:

--->JSON:

--->jQuery:

### 例子

<i class="fa fa-trash-o" onclick="del('${link.id}')"></i>

del = function (id){

var xhr=createXHR();

alert(id)

xhr.onreadystatechange=function()

{

if(xhr.readyState==4)

{

if(xhr.status>=200&&xhr.status<300||xhr.status==304)

{

alert("删除了")

window.location.reload();

//alert(xhr.responseText);

// doDeleteSuccess(id);

}

}

}

xhr.open("get","/link/delete?id="+id,true);

xhr.send(null);

}

## JSON数据格式

## 什么是JSON

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。它基于ECMAScript的一个子集。 JSON采用完全独立于语言的文本格式，但是也使用了类似于C语言家族的习惯（包括C、C++、C#、Java、JavaScript、Perl、Python等）。这些特性使JSON成为理想的数据交换语言。 易于人阅读和编写，同时也易于机器解析和生成(一般用于提升网络传输速率)。

JSON格式

​ JSON对象

{ key1:value}

{"username":"zhangsan","password":"123"}

​ JSON数组

[{ key1:value},{ key1:value},{ key1:value}]

​

使用BootStrap开发一个响应式的页面出来

需求分析

开发一套响应式页面.让他能够在各种设备上显示正常,提升用户体验

技术分析

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。

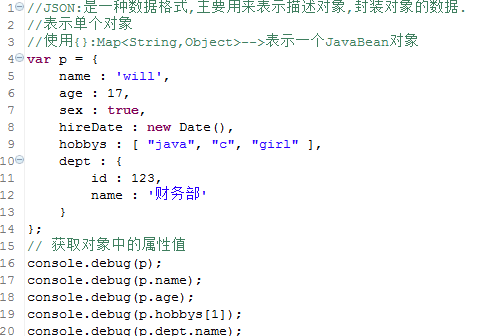
[

{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);

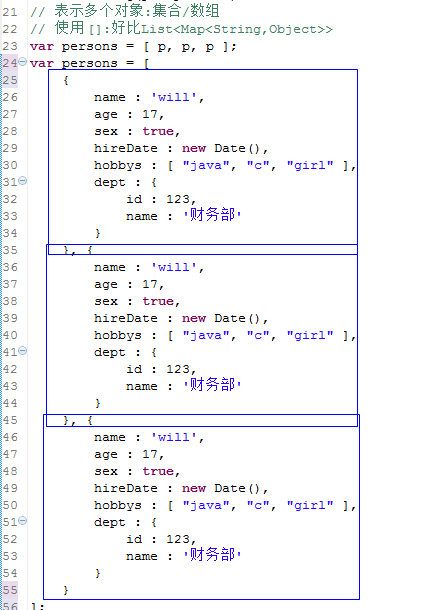
//eval函数,可以把一个字符串转换为JS代码

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**.

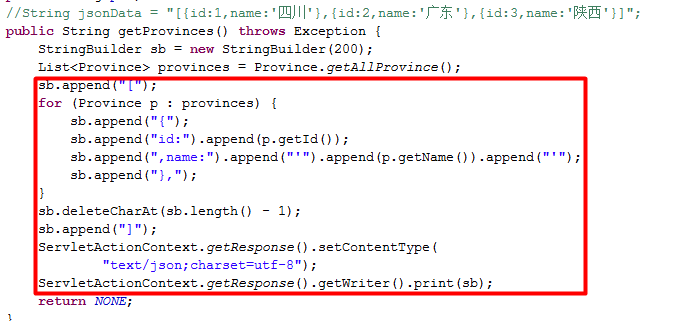
--------------------------------------------------------------------

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'}]**";



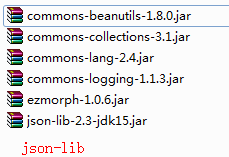


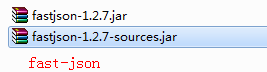
### JSON转换工具(JSON lib和fast JSON)

**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.)





### fastJsoN

1、//将map转换成jsonObject

JSONObject itemJSONObj = JSONObject.parseObject(JSON.toJSONString(itemMap));

将Map类型的itemInfo转换成json，再经JSONObject转换实现。  
  
2、//将jsonObj转换成Map  
Map<String, Object> itemMap = JSONObject.toJavaObject(itemJSONObj, Map.class);

//JOSN.parseObjet()方法同样可以转换

3、//将List转换成JSONArray

JSONArray ja = JSONArray.parseArray(JSON.toJSONString(itemList));

4、//将JSONArray转换成List

这个涉及功能比较广，有达人整理的更为完善。

链接如下：

http://www.cnblogs.com/goody9807/p/4244862.html

5 json转成对象

List<ChannelItem> channelItemList  = JSON.parseArray(itemJson,ChannelItem.class);

### Struts2的JSON插件

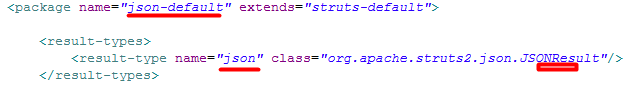
**Struts2本身是支持AJAX操作的,支持把对象转换为JSON数据格式**.

**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**.



### 转换

    result = JSONObject.fromObject(map).toString();//后台代码，将map转成json对象

           success : function(result) {//前台返回值代码  
var obj = eval("("+result+")");  
//var data1 =(new Function("","return "+obj))();  
var data1=eval("("+obj+")");  
alert(data1.name);

           ｝

## 获取参数

以下用法将以此代码段为例：

<form action='' method='' name='form1'>

<input type='text' name='content' id='content'>

</form>

### 用法一：html中直接传参

<input type='text' id='content' onBlur='check(this.value)'>

此用法是在input失去焦点时触发js的check(obj)函数，并将<当前input的数据传递过去。

### 用法二：通过form的name属性直接获取（js）

var txt = form1.content.value;

### 用法三：通过id和name并经由dom来取得

var txt = document.getElementById("content").value;

var txts = document.getElementByName("content").value;

## a标签跳转js代码

a href="javascript:js\_method();"

这是我们平台上常用的方法，但是这种方法在传递this等参数的时候很容易出问题，而且javascript:协议作为a的href属性的时候不仅会导致不必要的触发window.onbeforeunload事件，在IE里面更会使[gif动画](https://www.baidu.com/s?wd=gif%E5%8A%A8%E7%94%BB&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)图片停止播放。[W3C标准](https://www.baidu.com/s?wd=W3C%E6%A0%87%E5%87%86&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)不推荐在href里面执行javascript语句

a href="javascript:void(0);" onclick="js\_method()"这种方法是很多网站最常用的方法，也是最周全的方法，onclick方法负责执行js函数，而void是一个操作符，void(0)返回undefined，地址不发生跳转。而且这种方法不会像第一种方法一样直接将js方法暴露在浏览器的状态栏。

a href="javascript:;" onclick="js\_method()"这种方法跟跟2种类似，区别只是执行了一条空的[js代码](https://www.baidu.com/s?wd=js%E4%BB%A3%E7%A0%81&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)。

a href="#" onclick="js\_method()"这种方法也是网上很常见的代码，#是标签内置的一个方法，代表top的作用。所以用这种方法点击后网页后返回到页面的最顶端。

a href="#" onclick="js\_method();return false;"这种方法点击执行了js函数后return false，页面不发生跳转，执行后还是在页面的当前位置。此外还可以用js(jquery)单独给标签绑定[click事件](https://www.baidu.com/s?wd=click%E4%BA%8B%E4%BB%B6&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)。

## 常见问题

### 乱码

传递参数的时候 对参数进行编码priceName=”encodeURI(priceName)”，也可以用encodeURIComponent();

1. js文件中使用encodeURI()方法。  
     
   var url = "Validate.jsp?id=" + encodeURI(encodeURI(idField.value));     
     
   2.在后台中对传递的参数进行URLDecoder解码  
     
   String username = URLDecoder.decode(request.getParameter("id"),"UTF-8");

前端编码两次：encodeURI(encodeURI(要编码的中文),'')

后端解码：URLDecoder.decode(request.getParameter(对应的字段名称), "UTF-8")

或者把请求方式变为post也能解决这个问题，想后端通过http的方式调用远程的接口，可以把中文编码，对header中加入

httpGet.setHeader("Content-Type","application/text; charset=UTF-8");这样相当于告诉了接收方字符的类型和编码方式。

#### server.xml

在tomcat的server.xml中的

<Connector connectionTimeout="20000" port="8080" protocol="HTTP/1.1" redirectPort="8443" URIEncoding="utf-8"/>

这段中加上 URIEncoding="utf-8"

**前端发送带中文的数据：**

var searchText = "英语";

//通过URL传递：需要编码两次

searchText = encodeURI(searchText);

searchText = encodeURI(searchText);

$.ajax({

type: 'GET',

url: 'search.action' + "?searchText=" + searchText,

data: '',

contentType: 'text/json,charset=utf-8',

dataType: 'json',

success: function (data) {

}

});

## 前端的一些编码方法：

<script type="text/javascript">

//escape()不能直接用于URL编码，它的真正作用是返回一个字符的Unicode编码值。比如"春节"的返回结果是%u6625%u8282，escape()不对"+"编码 主要用于汉字编码。

alert(escape("春节"));

alert(unescape(escape("春节")));

//encodeURI()是用来对URL编码的函数。 编码整个url地址，但对特殊含义的符号"; / ? : @ & = + $ , #"不进行编码。对应的解码函数是：decodeURI()。

alert(encodeURI('http://baidu.com?hello=您好&word=文档'));

alert(decodeURI(encodeURI('http://baidu.com?hello=您好&word=文档')));

//encodeURIComponent() 能编码"; / ? : @ & = + $ , #"这些特殊字符。对应的解码函数是decodeURIComponent()。

alert(encodeURIComponent('http://baidu.com?hello=您好&word=文档'));

alert(decodeURIComponent(encodeURIComponent('http://baidu.com?hello=您好&word=文档')));

</script>

## js如何准确获取当前页面url网址信息

在WEB开发中，时常会用到javascript来获取当前页面的url网址信息，在这里是我的一些获取url信息的小总结。

下面我们举例一个URL，然后获得它的各个组成部分：http://i.cnblogs.com/EditPosts.aspx?opt=1

1、window.location.href(设置或获取整个 URL 为字符串)

var test = window.location.href;

alert(test);

返回：http://i.cnblogs.com/EditPosts.aspx?opt=1

2、window.location.protocol(设置或获取 URL 的协议部分)

var test = window.location.protocol;

alert(test);

返回：http:

3、window.location.host(设置或获取 URL 的主机部分)

var test = window.location.host;

alert(test);

返回：i.cnblogs.com

4、window.location.port(设置或获取与 URL 关联的端口号码)

var test = window.location.port;

alert(test);

返回：空字符(如果采用默认的80端口(update:即使添加了:80)，那么返回值并不是默认的80而是空字符)

5、window.location.pathname(设置或获取与 URL 的路径部分（就是文件地址）)

var test = window.location.pathname;

alert(test);

返回：/EditPosts.aspx

6、window.location.search(设置或获取 href 属性中跟在问号后面的部分)

var test = window.location.search;

alert(test);

返回：?opt=1

PS：获得查询（参数）部分，除了给动态语言赋值以外，我们同样可以给静态页面，并使用javascript来获得相信应的参数值。

7、window.location.hash(设置或获取 href 属性中在井号“#”后面的分段)

var test = window.location.hash;

alert(test);

返回：空字符(因为url中没有)

8、js获取url中的参数值

一、正则法

function getQueryString(name) {

var reg = new RegExp('(^|&)' + name + '=([^&]\*)(&|$)', 'i');

var r = window.location.search.substr(1).match(reg);

if (r != null) {

return unescape(r[2]);

}

return null;

}

// 这样调用：

alert(GetQueryString("参数名1"));

alert(GetQueryString("参数名2"));

alert(GetQueryString("参数名3"));

二、split拆分法

function GetRequest() {

var url = location.search; //获取url中"?"符后的字串

var theRequest = new Object();

if (url.indexOf("?") != -1) {

var str = url.substr(1);

strs = str.split("&");

for(var i = 0; i < strs.length; i ++) {

theRequest[strs[i].split("=")[0]] = unescape(strs[i].split("=")[1]);

}

}

return theRequest;

}

var Request = new Object();

Request = GetRequest();<br>// var id=Request["id"];

// var 参数1,参数2,参数3,参数N;

// 参数1 = Request['参数1'];

// 参数2 = Request['参数2'];

// 参数3 = Request['参数3'];

// 参数N = Request['参数N'];

三、指定取

比如说一个url：http://i.cnblogs.com/?j=js,我们想得到参数j的值，可以通过以下函数调用。

function GetQueryString(name) {

var reg = new RegExp("(^|&)" + name + "=([^&]\*)(&|$)", "i");

var r = window.location.search.substr(1).match(reg); //获取url中"?"符后的字符串并正则匹配

var context = "";

if (r != null)

context = r[2];

reg = null;

r = null;

return context == null || context == "" || context == "undefined" ? "" : context;

}

alert(GetQueryString("j"));

四、单个参数的获取方法

function GetRequest() {

var url = location.search; //获取url中"?"符后的字串

if (url.indexOf("?") != -1) { //判断是否有参数

var str = url.substr(1); //从第一个字符开始 因为第0个是?号 获取所有除问号的所有符串

strs = str.split("="); //用等号进行分隔 （因为知道只有一个参数 所以直接用等号进分隔 如果有多个参数 要用&号分隔 再用等号进行分隔）

alert(strs[1]); //直接弹出第一个参数 （如果有多个参数 还要进行循环的）

}

}

## 常见问题

异常信息

场景描述

<script type="text/javascript" src="${pageContext.request.contextPath }/dist/js/zy/index.js"></script>

异常原因

没有指定引入js文件的类型

解决方案

添加 script的type指定

<script type="text/javascript" src="${pageContext.request.contextPath }/dist/js/zy/index.js"></script>

参考资料

## button

### 跳转链接

单独添加一个链接，实现跳转：

<input type="button" value="我是一个按钮"

onclick="javascrtpt:window.location.href='http://blog.sina.com.cn/mleavs'">

触发一个函数跳转：

<script>

function jump(){

window.location.href="http://blog.sina.com.cn/mleavs";

}

</script>

<input type="button" value="我是一个按钮" onclick=javascrtpt:jump()>

# JQ

## 删除和隐藏

### 删除元素

删除id为div1的元素

$("#div1").remove();

删除 class="italic" 的所有 <p> 元素：

$("p").remove(".italic");

### 控制元素的隐藏和显示的几种方法。

使用jquery控制div的显示与隐藏，一句话就能搞定，例如：

1.$("#id").show()表示为display：block，

$("#id").hide()表示为display:none;

2.$("#id").toggle()切换元素的可见状态。如果元素是可见的，切换为隐藏的；如果元素是隐藏的，则切换为可见的。

3.$("#id").css('display','none');//隐藏

$("#id").css('display','block');//显示

或者

$("#id")[0].style.display='none';

　　display=none 控制对象的隐藏搜索

　　display=block控制对象的显示

4.$("#id").css('visibility','hidden');//元素隐藏

$("#id").css('visibility','visible');//元素显示

CSS visibility 属性规定元素是否可见。

visible　元素可见。

hidden　元素不可见。

collapse　在表格元素中使用时，此值可删除一行或一列，但它不影响表格的布局。被行或列占据的空间会留给其他内容使用。如果此值被用在其他的元素上，会呈现为 "hidden"。

inherit　从父元素继承 visibility 属性的值。

注意：

display:none和visible:hidden都能把网页上某个元素隐藏起来，在视觉效果上没有区别，但是在一些DOM操作中两者有区别:

display:none ---不为被隐藏的对象保留其物理空间，即该对象在页面上彻底消失，通俗来说就是看不见也摸不到。

visible:hidden--- 使对象在网页上不可见，但该对象在网页上所占的空间没有改变，即它仍然具有高度、宽度等属性，通俗来说就是看不见但摸得到。

//第1种方法 ,给元素设置style属性

$("#hidediv").css("display", "block");

//第2种方法 ,给元素换class，来实现隐藏div，前提是换的class样式定义好了隐藏属性

$("#hidediv").attr("class", "blockclass");

//第3种方法,通过jquery的css方法，设置div隐藏

$("#blockdiv").css("display", "none");

$("#hidediv").show();//显示div

$("#blockdiv").hide();//隐藏div

### 案例

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>jQuery便利多个相同的class，点击显示隐藏图标，显示及隐藏其下的内容</title>

<link rel="stylesheet" type="text/css" href="bootstrap/css/bootstrap.min.css">

<script type="text/javascript" src="bootstrap/js/jquery.min.js"></script>

<script type="text/javascript" src="bootstrap/js/bootstrap.min.js"></script>

</head>

<body>

<style>

body{

background-color: #FFFFFF;

letter-spacing: 2px;

}

.box{

padding-top: 100px;

width: 50%;

margin: 0 auto;

}

ul li ul{

margin: 20px 0 20px 40px;

}

.img\_sh{

margin: 20px 0;

}

.img\_sh img{

float: left;

padding-right: 10px;

}

</style>

<script>

//页面加载完毕后加载

window.onload = function () {

//遍历class类名叫img\_sh的li

$(".img\_sh").each(function () {

//将遍历到的类赋给li

var li = $(this);

//执行操作，li的子元素p标签里的子元素类名为jia的加号图片点击后执行函数

li.children("p").children(".jia").click(function () {

//点击的自身加号图片要隐藏

li.children("p").children(".jia").hide();

//点击完的加号图片要隐藏，而相反类名为jian的减号图片要显示，toggle的作用是取反（既显示或隐藏）

li.children("p").children(".jian").toggle();

//下面的ul列也是取反

li.children("ul").toggle();

})

//同理

li.children("p").children(".jian").click(function () {

li.children("p").children(".jian").hide();

li.children("p").children(".jia").toggle();

li.children("ul").toggle();

})

})

}

</script>

<div class="container-fluid">

<div class="box ">

<ul class="list-unstyled">

<li class="img\_sh">

<p>

刘吉祥jq测试

<img src="images/img01.jpg" alt="" class="img-responsive jia" style="display: none">

<img src="images/img02.jpg" alt="" class="img-responsive jian" style="display: block" >

</p>

<ul class="list-unstyled" style="display: block">

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

</ul >

</li>

<li class="img\_sh">

<p>

刘吉祥jq测试

<img src="images/img01.jpg" alt="" class="img-responsive jia" style="display: block" >

<img src="images/img02.jpg" alt="" class="img-responsive jian" style="display: none">

</p>

<ul class="list-unstyled" style="display: none">

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

</ul >

</li>

<li class="img\_sh">

<p>

刘吉祥jq测试

<img src="images/img01.jpg" alt="" class="img-responsive jia" style="display: block" >

<img src="images/img02.jpg" alt="" class="img-responsive jian" style="display: none">

</p>

<ul class="list-unstyled" style="display: none">

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

</ul >

</li>

<li class="img\_sh">

<p>

刘吉祥jq测试

<img src="images/img01.jpg" alt="" class="img-responsive jia" style="display: block" >

<img src="images/img02.jpg" alt="" class="img-responsive jian" style="display: none">

</p>

<ul class="list-unstyled" style="display: none">

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

</ul >

</li>

<li class="img\_sh">

<p>

刘吉祥jq测试

<img src="images/img01.jpg" alt="" class="img-responsive jia" style="display: block" >

<img src="images/img02.jpg" alt="" class="img-responsive jian" style="display: none">

</p>

<ul class="list-unstyled" style="display: none">

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

<li>●&nbsp 吃葡萄不吐葡萄皮，不吃葡萄倒吐葡萄皮;</li>

</ul >

</li>

</ul>

</div>

</div>

</body>

</html>

<https://www.jb51.net/article/145446.htm>

## 插件

<http://layer.layui.com/>

## 背景

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:

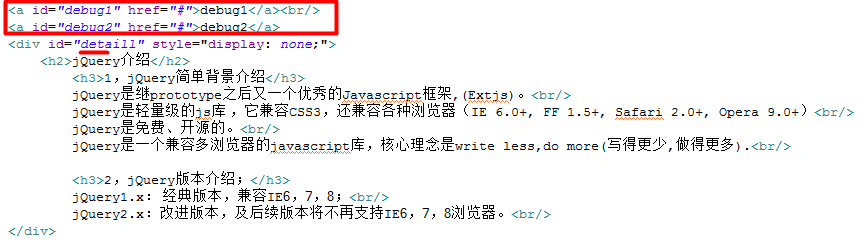
使用不同的jQuery插件,可能依赖的jQuery库的版本不一致,此时请保留插件依赖的jQuery库.

**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**

**});**

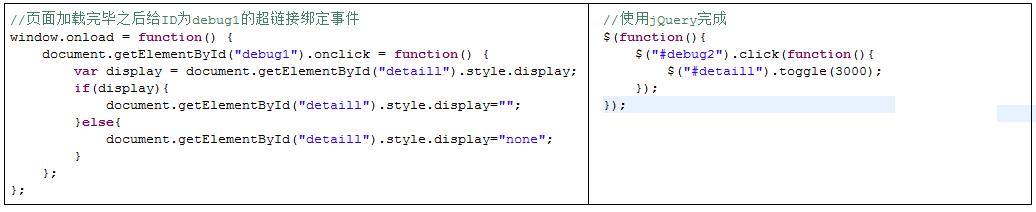
---------------------------------------------------------------------------

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.

## 表单验证

### jQuery控制表单提交

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>jQuery阻止表单提交.html</title>

<script type="text/javascript" src="../js2/jquery-1.7.2.js"></script>

<script type="text/javascript">

$(document).ready(function () {

$("#form").submit(function () {

// if("验证通过"){

// return true;

// }else{

// //验证失败

// return false;

// }

//window.location.href = "index.jsp";

})

})

</script>

</head>

<body>

<form action="postServlet5" method="post" id="form">

用户名：<input type="text" name="username" id="username"/><br>

密码：<input type="password" name="password" id="password"><br>

<input type="submit" value="submit"/>

</form>

</body>

</html>

### js检测密码强度

强度规则

任何在1-6之间的一类字符的组合，弱；例如： win

任何字符数的两类字符组合，中； 例如： win123

12位字符数以下的三类或四类字符组合，强 例如 win123abcABC

12位字符数以上的三类或四类字符组合，非常好。 例如：win123abcABC!

代码实现

function checkStrong(sValue) {

var modes = 0;

//正则表达式验证符合要求的

if (sValue.length < 1) return modes;

if (/\d/.test(sValue)) modes++; //数字

if (/[a-z]/.test(sValue)) modes++; //小写

if (/[A-Z]/.test(sValue)) modes++; //大写

if (/\W/.test(sValue)) modes++; //特殊字符

//逻辑处理

switch (modes) {

case 1:

return 1;

break;

case 2:

return 2;

case 3:

case 4:

return sValue.length < 12 ? 3 : 4

break;

}

}

### 插件

<https://www.runoob.com/jquery/jquery-plugin-validate.html>

## 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**.

### 例子

<button type="button" class="btn btn-primary" id="loginBut" onclick="addCart(this)">

提交更改

</button>

<script type="text/javascript">

$(function () {

$("#loginBut").click(function () {

// alert("111");

console.log('hello');

});

});

</script>

## 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



修改class

<div class="alert alert-danger alert-dismissable hide" role="alert">

<button type="button" class="close" data-dismiss="alert" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

<strong>这里是错误信息</strong>

</div>

$('.alert-danger').removeClass('hide').addClass('in')

## Ajax

### 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){

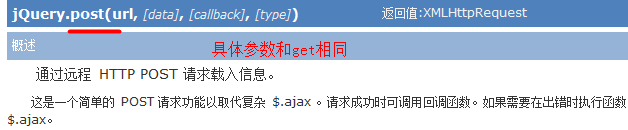
处理正确的响应操作

}

}

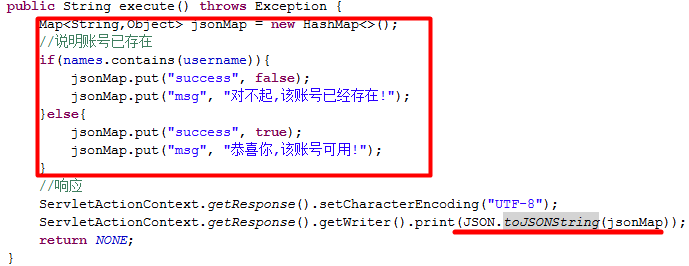
3:发送Ajax请求:

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**.

### 初始案例2

#### **$.ajax代码：**

<script src="../js/jquery.mobile-1.4.5.min.js"></script>

<script language="javascript">

function login(loginform){//传入表单参数

if(loginform.username.value==""){ //验证用户名是否为空

alert("请输入用户名！");loginform.username.focus();return false;

}

if(loginform.password.value==""){ //验证密码是否为空

alert("请输入密码！");loginform.password.focus();return false;

}

var param="/wechat/UserServlet?action=login&username="+loginform.username.value+"&password="+loginform.password.value; //将登录信息连接成字符串，作为发送请求的参数

$.ajax({

url:param,

type:"POST",

dataType:"json",

contentType: 'application/x-www-form-urlencoded; charset=UTF-8',//防止乱码

success:function(data){

if(data == false){

alert("您输入的用户名或密码有错！");loginform.username.focus();return false;

}else{

window.location.href = "index.html";//跳转到主页

}

}

});

}

</script>

#### html代码

<!DOCTYPE html>

<html>

<head>

<title>登陆测试</title>

<meta http-equiv="Content-Type" content="text/html;charset=utf-8">

<meta http-equiv="X-UA-Compatible" >

<meta name="renderer" content="webkit|ie-stand|ie-comp" />

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="../css/jquery.mobile-1.4.5.min.css">

<style>

a{margin:right;}

</style>

</head>

<body>

<div data-role="page" id="pageone" data-theme="a">

<div data-role="header">

<h1 style="font-family:'宋体',Arial, Helvetica, sans-serif;">测试登陆</h1>

</div>

<div data-role="main" class="ui-content">

<div class="ui-field-contain">

<form name="loginform" method="POST" id="loginform" action="">

<div class="ui-field-contain">

<input autocomplete="off" type="text" name="username" placeholder="用户名" data-rule="empty" data-name="帐号">

</div><br>

<div class="ui-field-contain">

<input autocomplete="off" type="password" name="password" placeholder="密码" data-rule="empty" data-name="密码">

</div><br>

<div class="ui-field-contain" align="right">

<a href="register.html" rel="external" float="right">没有帐号，立即注册</a>

</div>

<div class="ui-field-contain">

<input type="submit" class="submit-btn" id="J\_submit" value="登 录" onClick="login(this.form)">

</div>

</form>

</div>

</div>

</div>

</body>

</html>

#### java

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doPost(request,response);//如果提交方式为GET，跳转到dopost执行

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String action = request.getParameter("action");//获取操作

if("login".equals(action)){//用户登陆

this.login(request, response);

}

}

//用户登录

private void login(HttpServletRequest request,HttpServletResponse response) throws ServletException, IOException {

User f = new User();//这里我写了一个实体类，可以不用实体类，直接付给调用的函数参数即可

request.setCharacterEncoding("UTF-8");//防止乱码

f.setusername(request.getParameter("username"));

f.setpassword(request.getParameter("password"));

int r = userdao.login(f);//userdao是UserDao类的实例，在servlet的一开始生成

//r是返回的用户id，根据你自己生成的id，修改下方if判断条件

boolean flag = false;//默认登录不成功

if(r >= 0){//用户登陆成功

HttpSession session = request.getSession();

session.setAttribute("username", f.getusername());

session.setAttribute("userid", r);

session.setAttribute("identify", f.getidentify());

flag = true;

}else{

flag = false;

}

response.setContentType("text/html;charset=utf-8");

PrintWriter out = response.getWriter();

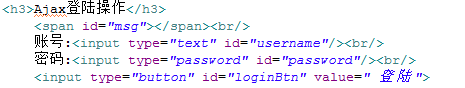
out.print(flag);//返回登录信息

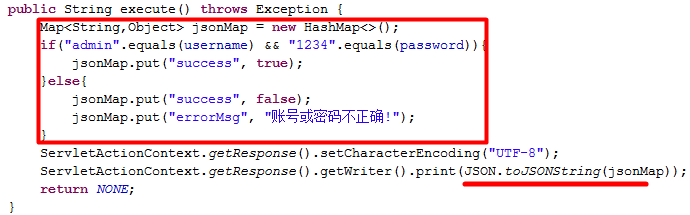
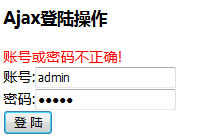
out.flush();

out.close();

}

### json案例:使用Ajax完成登陆操作.





### 返回json案例

ajax

addGroupAjax = function (parentGroupName,catId){

alert("进入复函数");

$.ajax({

url : '/group/add',

type : 'post',

async: true,//使用同步的方式,true为异步方式

data : {'groupName':parentGroupName, 'catId':catId},//这里使用json对象

contentType: 'application/x-www-form-urlencoded; charset=UTF-8',//防止乱码

success : function(data){

alert("成功啦");

$('#addGroup').modal('toggle');

window.location.reload();

},

fail:function(){

alert("。。。。。。。");

}

});

}

java

request.setCharacterEncoding("UTF-8");//防止乱码

int i = groupService.deleteByPrimaryKey(groupId);

HashMap<String, Object> map = new HashMap<>();

if (i>0){

//删除成功

map.put("success",true);

map.put("msg",i);

}else {

map.put("success",false);

map.put("msg","删除错误");

}

response.setContentType("text/html;charset=utf-8");

JSONObject jsonObject = JSON.parseObject(JSON.toJSONString(map));

PrintWriter out = response.getWriter();

//返回信息

out.print(jsonObject);

out.flush();

out.close();

### ajax提交返回boolean类型

1. 全局变量

在函数外定义window.flag=false;

ajax局部代码

if(data.success == true){

layer.tips('邮箱可用', '#userEml',{

tipsMore: true,

tips: [4, '#78BA32']

});

flag = true;

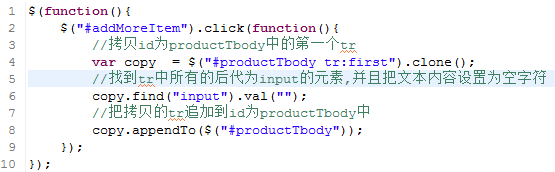
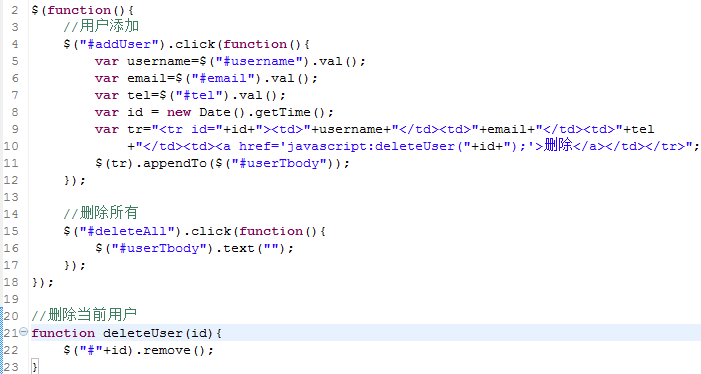
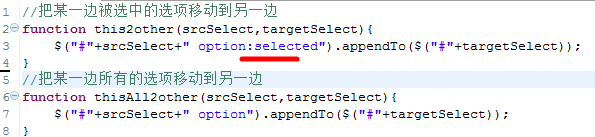
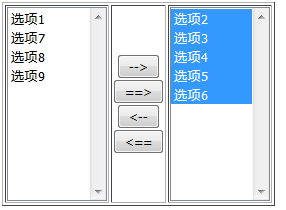
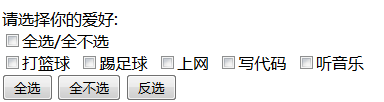
// alert(is);

2、在ajax方法外面定义临时遍历，接收boolean值也就是success中的值,这种方式需要同步才能接收到值因为ajax不会等这个方法运行完了才执行下面的代码的，可以用setTimeout延时下、

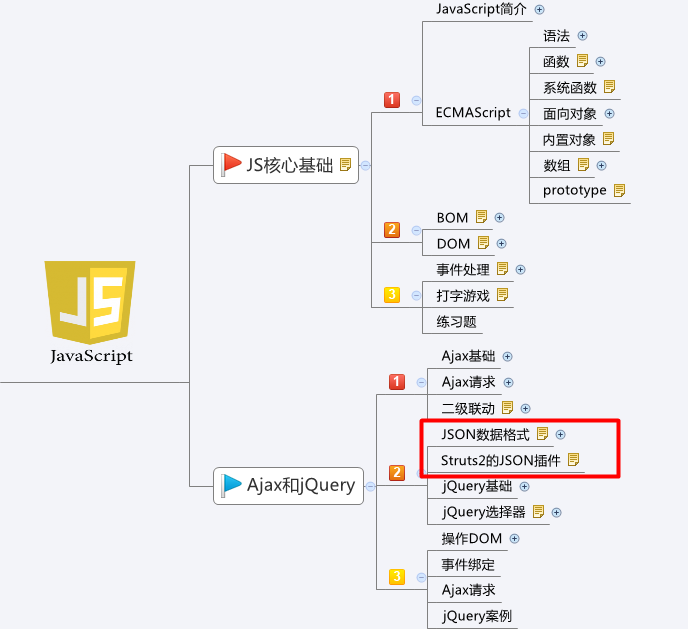
3、把你值放在$.dada()的缓存中保存；如：$("div").data("temp", true);alert($("div").data("temp"));

4、可以设置在属性中保存，然后取到。如果不想要了还可以删掉

## jQuery操作练习



## 小结



## 常见问题

### jQuery ajax post Uncaught RangeError: Maximum call stack size exceeded

异常信息

Maximum call stack size exceeded

jquery.min.js:4 Uncaught RangeError: Maximum call stack size exceeded

at RegExp.test (<anonymous>)

at Object.<anonymous> (jquery.min.js:4)

at Function.each (jquery.min.js:2)

at Ab (jquery.min.js:4)

at Ab (jquery.min.js:4)

at Ab (jquery.min.js:4)

at Ab (jquery.min.js:4)

at Function.r.param (jquery.min.js:4)

at Function.ajax (jquery.min.js:4)

at register:210

场景描述

注册时异步验证用户是否存在

var userEml = document.getElementById("userEml").val(); //要验证的对象

$.ajax({

url : '/user/chechEmailExist',

type : 'post',

async: false,//使用同步的方式,true为异步方式

data : {

'userEml':userEml

},//这里使用json对象

dataType : "json",

contentType: 'application/x-www-form-urlencoded; charset=UTF-8',//防止乱码

success : function(data){

if(data.success == true){

// layer.msg('邮箱可用', {icon: 1});

/\* layer.tips('邮箱可用', '#userEml',{

tipsMore: true,

tips: [4, '#78BA32']

});\*/

console.log("邮箱可用");

// return true;

}else{

// layer.msg('邮箱已存在', {icon: 2});

/\* layer.tips('邮箱已存在', '#userEml',{

tipsMore: true,

tips: [2, '#ff120a']

});\*/

console.log("邮箱已存在");

// return false;

}

},

error:function(){

layer.msg('error:请求失败', {icon: 2});

// return false;

},

}

);

异常原因

应该是类型转换的问题吧，获取值获取成地址之类的。

解决方案

var userEml = document.getElementById("userEml"); //要验证的对象

更改为

// var userEml = $("#userEml").val();

参考资料

<https://stackoverflow.com/questions/37883313/jquery-ajax-post-uncaught-rangeerror-maximum-call-stack-size-exceeded>