

arena teaching system

Ajax和JQuery



Java企业应用及互联网
高级工程师培训课程

达内集团教学研发部 编著

目 录

Unit01	1
1. Ajax	3
1.1. AJAX 原理	3
1.1.1. 注册时存在的问题	3
1.1.2. 什么是 Ajax	3
1.1.3. AJAX 工作原理	3
1.1.4. 如何获得 AJAX 对象	4
1.2. Ajax 对象属性和方法	4
1.2.1. 异步对象的属性和方法	4
1.2.2. onreadystatechange	5
1.2.3. readyState	5
1.3. 使用 Ajax 发送异步请求	5
1.3.1. 发送异步请求的步骤	5
1.3.2. 获取 AJAX 对象	6
1.3.3. 编写回调事件处理函数	6
1.3.4. 创建请求-GET 请求	6
1.3.5. 创建请求-POST 请求	7
1.3.6. 发送请求	7
1.3.7. 编写服务器端代码	7
1.3.8. Ajax 的应用	8
2. 编码问题	8
2.1. POST 请求时的乱码问题	8
2.1.1. 乱码产生的原因	8
2.1.2. 解决方式	8
2.2. GET 请求时的乱码问题	9
2.2.1. 为什么会产生乱码	9
2.2.2. 解决 GET 请求时的乱码问题	9
2.2.3. Ajax 的典型应用	10
经典案例	11
1. 获得 Ajax 对象	11
2. 使用 Ajax 对象发送 GET 请求	13
3. 使用 Ajax 对象发送 POST 请求	16
4. 使用 Ajax 校验用户名	19
5. POST 请求时的乱码处理	23
6. GET 请求时的乱码处理	26

7. 使用 Ajax 实现级联的下拉列表	29
课后作业	35
Unit02	36
1. JSON	37
1.1. JSON 简介	37
1.1.1. 什么是 JSON	37
1.1.2. 与平台无关的数据交换	37
1.1.3. 轻量级的解决方案	37
1.2. JSON 语法	38
1.2.1. JSON 的结构	38
1.2.2. 使用 JSON 表示一个对象	38
1.2.3. 使用 JSON 表示一个数组	38
2. 使用 JSON 实现数据交换	39
2.1. 数据交换	39
2.1.1. 数据交换原理	39
2.1.2. JSON 字符串-->JavaScript 对象	39
2.1.3. Java 对象转换前的准备	40
2.1.4. Java 对象转换成 JSON	40
2.2. 缓存问题	41
2.2.1. 什么是缓存问题	41
2.2.2. 如何解决缓存问题	41
2.3. 同步问题	42
2.3.1. 什么是同步问题	42
2.3.2. 如何发送同步请求	42
经典案例	43
1. 访问 JavaScript 对象的属性	43
2. 访问 Javascript 对象数组	44
3. JSON 字符串转换成 JavaScript 对象	46
4. Java 对象转换成 JSON 字符串	49
5. 使用 JSON 完成级联下拉列表	54
6. 热卖商品动态显示	59
课后作业	67
Unit03	68
1. jQuery	70
1.1. jQuery 介绍	70
1.1.1. 什么是 jQuery	70
1.2. jQuery 的编程步骤	70
1.2.1. 浏览器如何解析 HTML 文档	70
1.2.2. 利用选择器定位节点	71

1.2.3. 调用方法操作节点	71
1.2.4. jQuery 的编程步骤	71
1.3. jQuery 对象与 DOM 对象之间的转换	72
1.3.1. 什么是 jQuery 对象	72
1.3.2. DOM 对象 -> jQuery 对象	72
1.3.3. jQuery 对象 -> DOM 对象	72
2. jQuery 选择器	73
2.1. jQuery 选择器	73
2.1.1. 什么是 jQuery 选择器	73
2.1.2. 选择器的种类	73
2.2. 基本选择器	73
2.2.1. #id	73
2.2.2. .class	74
2.2.3. element	74
2.2.4. selector1,selector2,...selectorN	74
2.2.5. *	75
2.3. 层次选择器	75
2.3.1. select1 空格 select2	75
2.3.2. select1 > select2	75
2.3.3. select1+select2	76
2.3.4. select1~select2	76
2.4. 过滤选择器	76
2.4.1. 基本过滤选择器	76
2.4.2. 内容过滤选择器	77
2.4.3. 可见性过滤选择器	78
2.4.4. 属性过滤选择器	78
2.4.5. 子元素过滤选择器	79
2.4.6. 表单对象属性过滤选择器	79
2.5. 表单选择器	80
2.5.1. 表单选择器	80
3. jQuery 操作 DOM	80
3.1. jQuery 操作 DOM-查询	80
3.1.1. html ()	80
3.1.2. text ()	80
3.1.3. val ()	81
3.1.4. attr ()	81
3.2. jQuery 操作 DOM-创建、插入、删除	81
3.2.1. 创建 DOM 节点	81
3.2.2. 插入 DOM 节点	82
3.2.3. 删除 DOM 节点	82
3.3. JQuery 操作 DOM - 复制节点	82

3.3.1. 复制 DOM 节点	82
3.4. JQuery 操作 DOM - 样式操作	83
3.4.1. 样式操作	83
3.5. JQuery 操作 DOM - 遍历节点	83
3.5.1. 遍历节点	83
经典案例	84
1. JQuery 简单案例	84
2. JQuery 对象与 DOM 对象的转换	86
3. JQuery 选择器	88
4. JQuery 操作 DOM	110
课后作业	125
Unit04	126
1. JQuery 事件处理	128
1.1. 事件处理	128
1.1.1. 使用 JQuery 实现事件绑定	128
1.1.2. 获得事件对象 event	128
1.1.3. 事件对象的常用属性	128
1.2. 事件冒泡	129
1.2.1. 什么是事件冒泡	129
1.2.2. 如何取消事件冒泡	129
1.3. 合成事件	129
1.3.1. JQuery 的合成事件种类	129
1.4. 模拟操作	130
1.4.1. 模拟操作的语法	130
2. JQuery 增强操作	130
2.1. JQuery 动画	130
2.1.1. 显示、隐藏的动画效果	130
2.1.2. 上下滑动式的动画实现	130
2.1.3. 淡入淡出式动画效果	131
2.1.4. 自定义动画效果	131
2.2. JQuery 类数组	131
2.2.1. 什么是类数组	131
2.2.2. 类数组的操作	132
2.3. JQuery 对 AJAX 的支持	132
2.3.1. load ()	132
2.3.2. get ()	133
2.3.3. ajax ()	133
经典案例	134

1. jQuery 处理事件	134
2. 事件冒泡	137
3. 合成事件	140
4. 模拟操作	143
5. jQuery 动画	145
6. jQuery 遍历操作	150
7. jQuery 对 Ajax 的支持	153
课后作业	163

AJAX 和 jQuery

Unit01

知识体系.....Page 3

Ajax	AJAX 原理	注册时存在的问题
		什么是 Ajax
		AJAX 工作原理
		如何获得 AJAX 对象
	Ajax 对象属性和方法	异步对象的属性和方法
		onreadystatechange
		readyState
	使用 Ajax 发送异步请求	发送异步请求的步骤
		获取 AJAX 对象
		编写回调事件处理函数
		创建请求-GET 请求
		创建请求-POST 请求
		发送请求
		编写服务器端代码
		Ajax 的应用
编码问题	POST 请求时的乱码问题	乱码产生的原因
		解决方式
	GET 请求时的乱码问题	为什么会产生乱码
		解决 GET 请求时的乱码问题
		Ajax 的典型应用

经典案例.....Page 11

获得 Ajax 对象	如何获得 AJAX 对象
使用 Ajax 对象发送 GET 请求	创建请求-GET 请求
使用 Ajax 对象发送 POST 请求	创建请求-POST 请求
使用 Ajax 校验用户名	编写回调事件处理函数
	发送请求
	编写服务器端代码
	Ajax 的应用

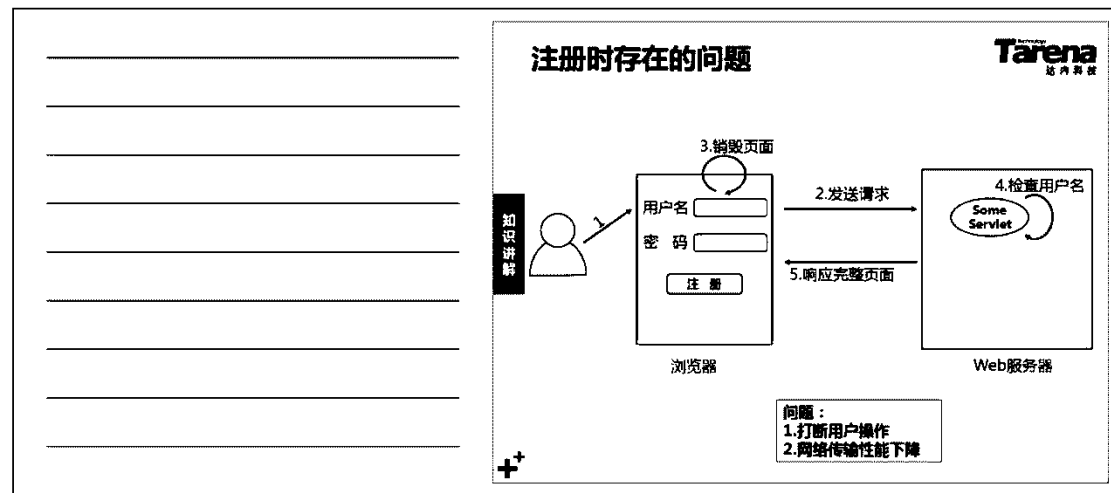
POST 请求时的乱码处理	乱码产生的原因
	解决方式
GET 请求时的乱码处理	为什么会产生乱码
	解决 GET 请求时的乱码问题
使用 Ajax 实现级联的下拉列表	Ajax 典型应用

课后作业.....Page 35

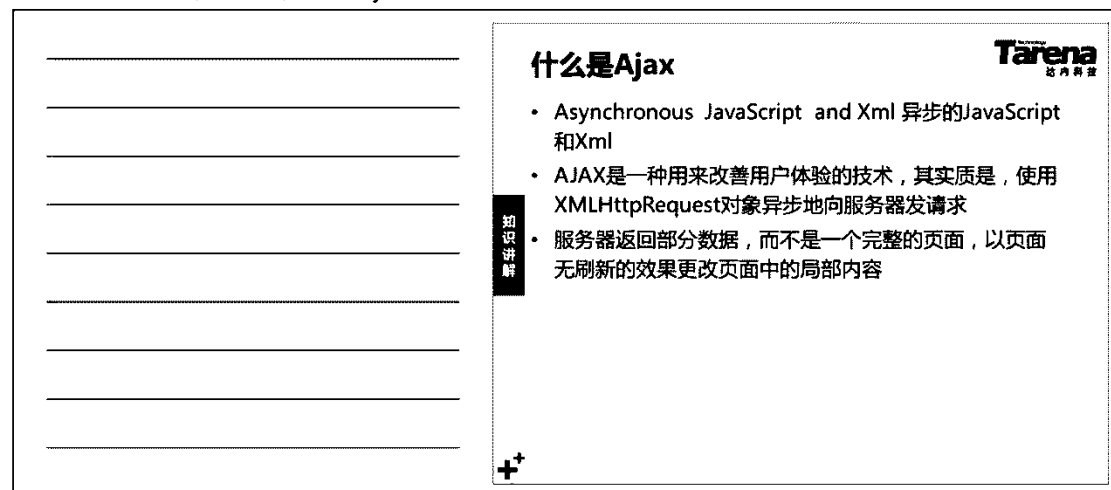
1. Ajax

1.1. AJAX 原理

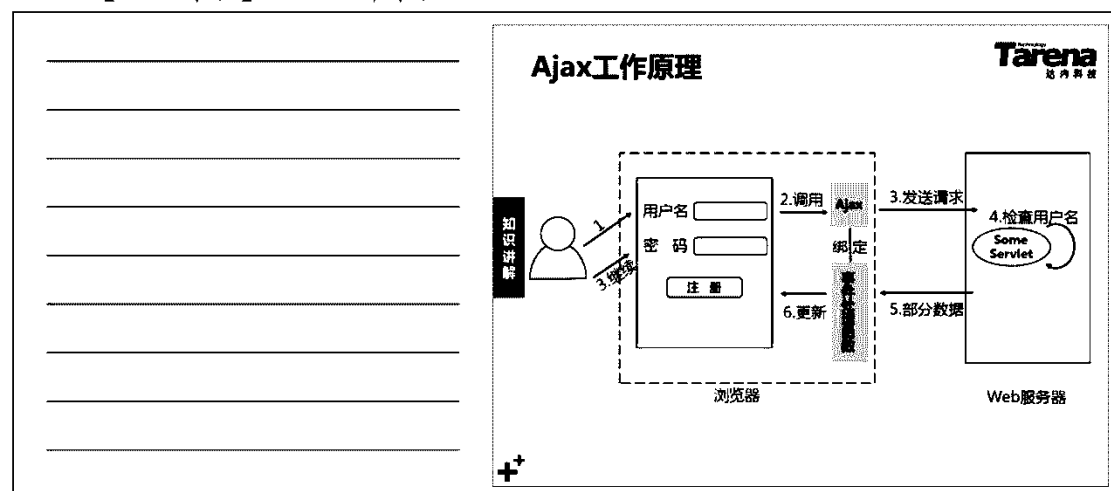
1.1.1. 【AJAX 原理】注册时存在的问题




1.1.2. 【AJAX 原理】什么是 Ajax



1.1.3. 【AJAX 原理】AJAX 工作原理



1.1.4. 【AJAX 原理】如何获得 AJAX 对象



如何获得Ajax对象

```


function getXhr(){
    var xhr = null;
    if(window.XMLHttpRequest){
        xhr = new XMLHttpRequest();
    }else{
        xhr = new
            ActiveXObject('Microsoft.XMLHttp');
    }
    return xhr;
}
        
```

++

知识讲解

1.2. Ajax 对象属性和方法

1.2.1. 【Ajax 对象属性和方法】异步对象的属性和方法




异步对象的属性和方法

属性/方法	说 明
abort ()	取消请求
getAllResponseHeaders ()	获取响应的所有Http头
getResponseHeader ()	获取指定的Http头
open (method,url)	创建请求, method请求类型 get post
send ()	发送请求
setRequestHeader ()	指定请求的Http头
onreadystatechange	发生任何状态变化时的事件控制对象
readyState	请求的状态 0 尚未初始化 1正在发送请求 2请求完成 3请求成功, 正在接受数据 4数据接收成功

++

知识讲解





异步对象的属性和方法 (续)

属性/方法	说 明
responseText	服务器返回的文本
responseXML	服务器返回的xml, 可以当做DOM处理
status	服务器返回的http请求响应值常用的有: 200 表示请求成功 202 请求被接受但处理未完成 400 错误的请求 404 资源未找到 500 内部服务器错误, 如asp代码错误等

++

知识讲解

1.2.2. 【Ajax 对象属性和方法】onreadystatechange



<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>onreadystatechange</h4> <ul style="list-style-type: none"> • onreadystatechange：绑定一个事件处理函数，该函数用来处理readystatechange事件。 • 注：当Ajax对象的readyState的值发生了改变，比如，从0变成了1，就会产生readystatechange事件 <div style="text-align: right;">  </div> <div style="text-align: right;">+</div>
---	--

1.2.3. 【Ajax 对象属性和方法】readyState

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>readyState</h4> <ul style="list-style-type: none"> • readyState：一共有5个值，分别是0,1,2,3,4，分别表示Ajax对象与服务器通信的状态 • 比如，当值为4时，表示Ajax对象已经获得了服务器返回的所有数据。 <div style="text-align: right;">  </div> <div style="text-align: right;">+</div>
---	---

1.3. 使用 Ajax 发送异步请求

1.3.1. 【使用 Ajax 发送异步请求】发送异步请求的步骤


<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>发送异步请求的步骤</h4> <ol style="list-style-type: none"> ① 获取Ajax对象：获取XMLHttpRequest对象实例 ② 设置回调函数：为Ajax对象的 readyStatechange事件设定响应函数 ③ 创建请求：调用XMLHttpRequest对象的open方法 ④ 发送请求：调用Ajax对象的send方法 <div style="text-align: right;">  </div> <div style="text-align: right;">+</div>
---	--

1.3.2. 【使用 Ajax 发送异步请求】获取 AJAX 对象




① 获取Ajax对象

```
var xhr = getXhr ( ) ;
function getXhr(){
    var xhr = null;
    if(window.XMLHttpRequest){
        xhr = new XMLHttpRequest();
    }else{
        xhr = new ActiveXObject('Microsoft.XMLHTTP');
    }
    return xhr;
}
```




1.3.3. 【使用 Ajax 发送异步请求】编写回调事件处理函数




② 编写回调事件处理函数

```
xhr.onreadystatechange = function(){
    if(xhr.readyState == 4 && xhr.status == 200 ){
        var txt = xhr.responseText;
        //DOM操作
    }
}
```



1.3.4. 【使用 Ajax 发送异步请求】创建请求-GET 请求



③ 创建请求-GET请求

```
xhr.open ( 'get' , 'xx.do' , true ) ;
```

• 注意：

- true：表示发送异步请求（当Ajax对象发请求时，用户仍然可以对当前页面做其它的操作）。
- false：表示发送同步请求（当Ajax对象发请求时，浏览器会锁定当前页面，用户不能对当前页面做其它操作）。

1.3.5. 【使用 Ajax 发送异步请求】创建请求-POST 请求

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">Tarena 达内科技</div> <h4>③ 创建请求-POST请求</h4> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <pre>xhr.open('post', 'xx.do', true); xhr.setRequestHeader('content-type', 'application/x-www-form-urlencoded');</pre> </div> <div style="display: flex; align-items: center;"> <div style="background-color: black; color: white; padding: 2px 5px; writing-mode: vertical-rl; margin-right: 5px;">知识讲解</div> <ul style="list-style-type: none"> • setRequestHeader的作用：因为HTTP协议要求发送post请求时，必须有content-type消息头，但是默认情况下xhr(即Ajax对象)不会添加该消息头，所以，需要调用setRequestHeader方法，添加这个消息头 </div> <div style="text-align: right;">++</div>
---	---



1.3.6. 【使用 Ajax 发送异步请求】发送请求

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">Tarena 达内科技</div> <h4>④ 发送请求</h4> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <pre>GET 请求：xhr . send(null) POST 请求：xhr . send(name=value & name=value...)</pre> </div> <div style="display: flex; align-items: center;"> <div style="background-color: black; color: white; padding: 2px 5px; writing-mode: vertical-rl; margin-right: 5px;">知识讲解</div> <ul style="list-style-type: none"> • GET请求： <ul style="list-style-type: none"> – send方法内传递null – 若要提交数据，则在open方法的“URL”后面追加 – 如：xhr . open("get" , " xx.do?name=value&name=value" , true) </div> <div style="text-align: right;">++</div>
---	--

1.3.7. 【使用 Ajax 发送异步请求】编写服务器端代码

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">Tarena 达内科技</div> <h4>编写服务器端代码</h4> <ul style="list-style-type: none"> • 服务器返回的一般是部分数据，比如一个简单的文本。 <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <pre>public void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { request.setCharacterEncoding("utf-8"); response.setContentType("text/html;charset=utf-8"); PrintWriter out = response.getWriter(); out.println("用户名已经存在"); }</pre> </div> <div style="display: flex; align-items: center;"> <div style="background-color: black; color: white; padding: 2px 5px; writing-mode: vertical-rl; margin-right: 5px;">代码标注</div> </div> <div style="text-align: right;">++</div>
---	---



1.3.8. 【使用 Ajax 发送异步请求】Ajax 的应用

<div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div>	<div style="text-align: right;"></div> <h4>Ajax的应用</h4> <ul style="list-style-type: none">• 输入的值需要校验，如检测注册的用户名是否已被占用• 搜索时出现的自动提示列表• 级联显示• 数据录入和列表显示在同一个页面• 不需要刷新的翻页 <div style="text-align: right;"></div> <div style="text-align: right;">+</div>
---	--



2. 编码问题

2.1. POST 请求时的乱码问题

2.1.1. 【POST 请求时的乱码问题】乱码产生的原因


<div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div>	<div style="text-align: right;"></div> <h4>乱码产生的原因</h4> <ul style="list-style-type: none">• 乱码问题产生的原因<ul style="list-style-type: none">– 所有浏览器提供的AJAX对象对请求参数使用UTF-8进行编码– 服务器默认使用iso-8859-1去解码– 编码与解码不同就会产生乱码 <div style="text-align: right;"></div> <div style="text-align: right;">+</div>
---	---

2.1.2. 【POST 请求时的乱码问题】解决方式


<div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 20px; margin-bottom: 5px;"></div>	<div style="text-align: right;"></div> <h4>解决方式</h4> <ul style="list-style-type: none">• request.setCharacterEncoding (“UTF-8”) ;• 注：火狐就不用这句代码，是因为这个浏览器会在发送的请求数据包中告诉服务器，它是哪种方式进行的数据编码。 <div style="text-align: right;"></div> <div style="text-align: right;">+</div>
---	--


2.2. GET 请求时的乱码问题

2.2.1. 【GET 请求时的乱码问题】为什么会产生乱码

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>为什么会产生乱码</h4> <ul style="list-style-type: none"> • IE浏览器提供的Ajax对象会使用GBK字符集对请求参数进行编码，而其它浏览器会使用UTF-8来编码。 • 服务器默认情况下会使用ios-8859-1进行解码。 • 编码与解码不一致即产生乱码 <div style="text-align: right;">+</div>
---	--

2.2.2. 【GET 请求时的乱码问题】解决 GET 请求时的乱码问题

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>解决GET请求时的乱码问题</h4> <ul style="list-style-type: none"> • step1 , <ul style="list-style-type: none"> – 指定字符集进行解码 – 比如，tomcat可以修改conf/server.xml文件中 <code><Connector URIEncoding="utf-8"></code> ,使得tomcat按utf-8方式解码。 <div style="text-align: right;">+</div>
---	---

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>解决GET请求时的乱码问题（续）</h4> <ul style="list-style-type: none"> • Step2 <ul style="list-style-type: none"> – 使用encodeURIComponent对请求地址进行编码 – encodeURIComponent会使用utf-8对请求地址中的中文参数进行编码 – 其实问题的根本原因就是IE的与众不同。修改完成后，重启tomcat，不用IE来运行就会发现可以正常读取表单的get方式提交的中文。 – 针对 IE <pre>var uri = 'xxx.do?uname='+document. getElementById('username').value; xhr.open('get ', encodeURIComponent(uri), true);</pre> <div style="text-align: right;">+</div>
---	--

2.2.3. 【GET 请求时的乱码问题】Ajax 的典型应用

经典案例

1. 获得 Ajax 对象

- 问题

如何获得 XMLHttpRequest 对象。

- 方案

区分浏览器，使用不同的获取方式。

- 步骤

步骤一：新建 ajax01.html 页面

新建一个 Web 工程，在 WebRoot 下新建 ajax01.html 页面。在<script>标记内编写 JavaScript 代码实现获取 Ajax 对象。如图-1 所示：

```
<html>
  <head>
    <title>ajax01.html</title>
    <meta http-equiv="Content-Type"
      content="text/html; charset=UTF-8">
    <script type="text/javascript">
      /*获取Ajax对象*/
      function getXhr(){
        var xhr = null;
        if(window.XMLHttpRequest){
          xhr = new XMLHttpRequest();
        }else{
          xhr = new ActiveXObject("Microsoft.XMLHttp");
        }
        return xhr;
      }
    </script>
  </head>
  <body>
    <!-- 获取XMLHttpRequest对象 -->
    <a href="javascript:;" onclick="alert(getXhr());">
      获取Ajax对象
    </a>
  </body>
</html>
```

图 - 1

步骤二：部署项目访问 ajax01.html 页面

访问页面如图-2 所示：



图 - 2

点击链接：



图 - 3

使用 IE 测试结果如图-4 所示：



图 - 4

注意：本次测试使用的是 IE11 ,但 IE 的早期 5.5 ,6.0 版本创建的都是 ActiveXObject 类型的。从 IE7 开始支持创建 XMLHttpRequest 的方式来获取 Ajax 对象。

- **完整代码**

ajax01.html 文件代码如下：

```
<html>
<head>
  <title>ajax01.html</title>
  <meta http-equiv="Content-Type"
    content="text/html; charset=UTF-8">
  <script type="text/javascript">
    /*获取Ajax 对象*/
    function getXhr(){
      var xhr = null;
      //确保 IE7,IE8 , FireFox 下可以运行
```

```

if(window.XMLHttpRequest){
    xhr = new XMLHttpRequest();
}else{
    //确保 IE6 可以运行，无视更古老的 IE 浏览器
    xhr = new ActiveXObject("Microsoft.XMLHttp");
}
return xhr;
//更简易的写法
//xhr = window.XMLHttpRequest?
//    new XMLHttpRequest():new ActiveXObject("Microsoft.XMLHttp");
}
</script>
</head>
<body>
    <!-- 获取 XMLHttpRequest 对象 -->
    <a href="javascript:;" onclick="alert(getXhr());">
        获取 Ajax 对象
    </a>
</body>
</html>

```

2. 使用 Ajax 对象发送 GET 请求

• 问题

使用 Ajax 对象发送 GET 类型的请求，从服务器端获取一小段文本。

• 方案

遵循创建 Ajax 对象，创建请求，设置回调函数，发送请求的步骤来完成一次异步请求。

• 步骤

步骤一：新建 ajax02.html 页面

```

<html>
<head>
    <title>ajax02.html</title>
    <meta http-equiv="Content-Type"
        content="text/html; charset=UTF-8">
    <script type="text/javascript">
        function getXhr(){
            var xhr = null;
            if(window.XMLHttpRequest){
                xhr = new XMLHttpRequest();
            }else{
                xhr = new ActiveXObject("Microsoft.XMLHttp");
            }
            return xhr;
        }
    </script>
</head>
<body>
    <!-- 获取服务器端一小段文本 -->
    <a href="javascript:;" onclick="_____">获取文本</a>
</body>
</html>

```

准备添加响应方法

图 - 5

步骤二：添加 `getText()` 方法，并补充超链接对该方法的调用

```
function getText(){
    //1. 获取Ajax对象
    var xhr = getXhr();
    //2. 设置回调函数
    xhr.onreadystatechange=function(){
        if(xhr.readyState==4 && xhr.status==200){
            var txt = xhr.responseText;
            alert(txt);
        }
    };
    //3. 创建请求
    xhr.open("get", "get_text.do", true);
    //4. 发送请求
    xhr.send(null);
}
```

图 - 6

步骤三：编写 `ActionServlet` 类

```
public void service(HttpServletRequest request,
                    HttpServletResponse response)
                    throws ServletException, IOException {

    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    //获取请求资源路径
    String uri = request.getRequestURI();
    String path = uri.substring(
        uri.lastIndexOf("/"),
        uri.lastIndexOf("."));
    if(path.equals("/get_text")){//get请求
        out.println("来自星星的你");
    }
    out.close();
}
```

图 - 7

步骤四：部署、运行，查看结果

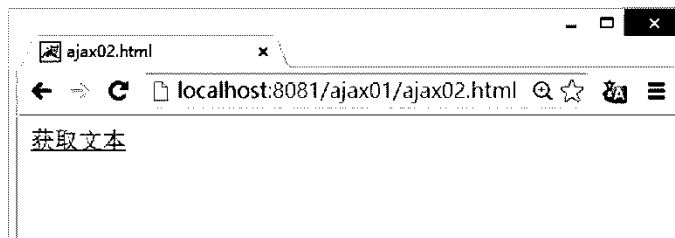


图 - 8

点击链接：

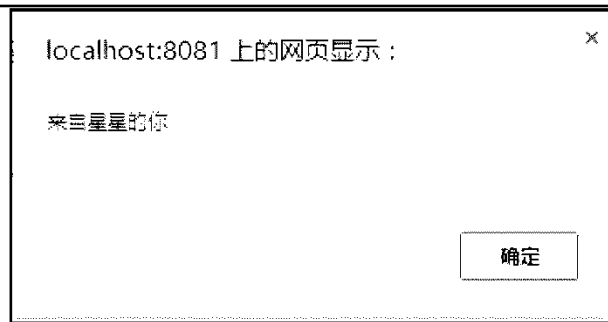


图 - 9

• 完整代码

ajax02.html 页面代码：

```
<html>
<head>
<title>ajax02.html</title>
<meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
<script type="text/javascript">
function getXhr(){
var xhr = null;
if(window.XMLHttpRequest){
xhr = new XMLHttpRequest();
}else{
xhr = new ActiveXObject("Microsoft.XMLHttp");
}
return xhr;
}
function getText(){
//1.获取 Ajax 对象
var xhr = getXhr();
//2.设置回调函数
xhr.onreadystatechange=function(){
if(xhr.readyState==4 && xhr.status==200){
var txt = xhr.responseText;
alert(txt);
}
};
//3.创建请求
xhr.open("get","get_text.do",true);
//4.发送请求
xhr.send(null);
}
</script>
</head>
<body>
<!-- 获取服务器端一小段文本 -->
<a href="javascript:;" onclick="getText()">获取文本</a>
</body>
</html>
```

ActionServlet.java 页面代码：

```
package web;
```

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ActionServlet extends HttpServlet {

    public void service(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //获取请求资源路径
        String uri = request.getRequestURI();
        String path = uri.substring(
            uri.lastIndexOf("/"),
            uri.lastIndexOf("."));
        if(path.equals("/get_text")){//get 请求
            out.println("来自星星的你");
        }
        out.close();
    }
}
```

web.xml 文件代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
    <display-name></display-name>
    <servlet>
        <servlet-name>ActionServlet</servlet-name>
        <servlet-class>web.ActionServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ActionServlet</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>
</web-app>
```

3. 使用 Ajax 对象发送 POST 请求

- 问题

使用 Ajax 对象发送 POST 类型的请求，并向服务器提交一小段文本，显示从服务器端返回的文本。

- 方案

遵循 GET 请求方式的 4 步基础上，多增加一个设定，设定请求头中的 content-type 属性的值为 application/x-www-form-urlencoded。

• 步骤

步骤一：新建 my.js 文件

在 WebRoot 节点下,新增一个文件夹“js”。在该文件夹中新增一个文件,名为 my.js。
将获取 XMLHttpRequest 对象的方法拷贝到该文件中,便于更多页面重用该方法。

```
1 /*获取Ajax对象*/
2 function getXhr() {
3     var xhr = null;
4     if(window.XMLHttpRequest) {
5         xhr = new XMLHttpRequest();
6     }else{
7         xhr = new ActiveXObject("Microsoft.XMLHttp");
8     }
9     return xhr;
10 }
```

图 - 10

步骤二：新建 ajax03.html 页面

```
<html>
<head>
  <title>ajax03.html</title>
  <meta http-equiv="Content-Type"
        content="type=text/html; charset=UTF-8">
  <script type="text/javascript" src="js/my.js"></script>
  <script type="text/javascript">
    function getText() {
      //发送post类型的请求
    }
  </script>
</head>
<body>
  <!-- 使用Ajax对象发送POST类型请求, 并获取文本 -->
  <input type="button" onclick="getText()" value="POST请求"/>
</body>
</html>
```

图 - 11

步骤三：编写 getText () 方法

```
function getText() {
  var xhr = getXhr();
  xhr.onreadystatechange=function() {
    if(xhr.readyState==4 && xhr.status==200) {
      alert(xhr.responseText);
    }
  };
  xhr.setRequestHeader('content-type',
    'application/x-www-form-urlencoded');
  xhr.open("post","post_text.do",true);
  xhr.send("uname=Dr.");
}
```

图 - 12

步骤四：修改 ActionServlet

```
if (path.equals("/get_text")) { //get请求
    out.println("来自星星的你");
} else if (path.equals("/post_text")) { //post请求
    String name = request.getParameter("uname");
    System.out.println(name);
    out.println("又来了一次的" + name);
}
```

图 - 13

步骤五：运行，查看结果



图 - 14

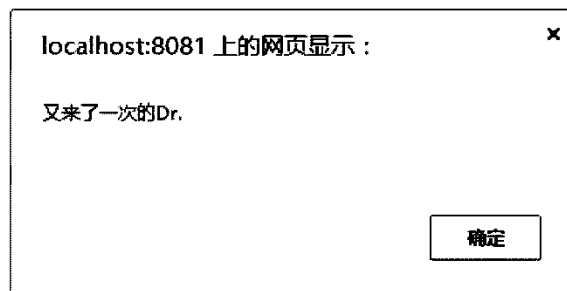


图 - 15

• 完整代码

ajax03.html 文件代码如下：

```
<html>
<head>
    <title>ajax03.html</title>
    <meta http-equiv="Content-Type"
        content="type=text/html; charset=UTF-8">
    <script type="text/javascript" src="js/my.js"></script>
    <script type="text/javascript">
        function getText() {
            var xhr = getXHR();
            xhr.onreadystatechange = function() {
                if (xhr.readyState == 4 && xhr.status == 200) {
                    alert(xhr.responseText);
                }
            };
            xhr.open("post", "post_text.do", true);
            xhr.setRequestHeader('content-type',
                'application/x-www-form-urlencoded');
            xhr.send("uname=Dr.");
        }
    </script>
</head>
<body>
    <button value="POST请求">
</body>
</html>
```

```
</script>
</head>
<body>
<!-- 使用 Ajax 对象发送 POST 类型请求，并获取文本 -->
<input type="button" onclick="getText()" value="POST 请求"/>
</body>
</html>
```

ActionServlet.java 文件代码如下：

```
package web;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ActionServlet extends HttpServlet {
    public void service(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");
        PrintWriter out = response.getWriter();
        //获取请求资源路径
        String uri = request.getRequestURI();
        String path = uri.substring(
            uri.lastIndexOf("/"),
            uri.lastIndexOf("."));
        if(path.equals("/get_text")){//get 请求
            out.println("来自星星的你");
        }else if(path.equals("/post_text")){//post 请求
            String name = request.getParameter("uname");
            System.out.println(name);
            out.println("又来了一次的" + name);
        }
        out.close();
    }
}
```

web.xml 文件代码参考案例 2。

my.js 文件代码参考图-10。

4. 使用 Ajax 校验用户名

• 问题

实现注册功能中，用户名是否可用的校验。

• 方案

填写完用户名后，通过点击后面的按钮进行验证。并显示服务器返回的说明信息。

- 步骤

步骤一：新建 regist.html 文件

如图-16 所示创建页面。

```
<body>
  <!-- 验证注册信息中的用户名是否可用 -->
  <form action="" method="post">
    <fieldset>
      <legend>注册信息</legend>
      用户名: <input name="uname" id="uname"/>
      <input type="button" value="检查一下吧"
        onclick="check_name()" />
      <span id="name_msg" style="color:red;"></span>
      <br/><br/>
      <input type="submit" value="注册"/>
    </fieldset>
  </form>
</body>
</html>
```

图 - 16

步骤二：编写 check_name () 方法

```
function check_name() {
  var xhr = getXHR();
  xhr.onreadystatechange=function() {
    if(xhr.readyState==4 && xhr.status==200) {
      document.getElementById("name_msg")
        .innerHTML = xhr.responseText;
    }
  };
  xhr.setRequestHeader('content-type',
    'application/x-www-form-urlencoded');
  xhr.open("post","check_name.do",true);
  document.getElementById("name_msg")
    .innerHTML="正在检查。。。";
  var uname = document.getElementById("uname");
  xhr.send("uname="+uname.value);
}
```

图 - 17

document.getElementById("name_msg").innerHTML= "正在检查 "；这行代码在服务器还没有返回信息之前，为用户提供良好的交互体验。

步骤三：修改 ActionServlet.java 文件

```
else if(path.equals("/check_name")){//检查用户名
    String name = request.getParameter("uname");
    //模拟一个耗时的操作
    if(1==1){
        try {
            Thread.sleep(6000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    System.out.println(name);
    if("Luffy".equals(name)){
        out.println("该用户名不可用");
    }else{
        out.println("可以使用!");
    }
}
```

模拟网络延时，出现“正在检查”的字样

图 - 18

步骤四：运行，查看结果

输入不可用的用户名：



图 - 19

提示结果如图-20：



图 - 20

输入可用的用户名如图-21 所示：



图 - 21

完整代码

regist.html 文件代码如下：

```
<html>
<head>
  <title>regist.html</title>
  <meta http-equiv="content-type"
    content="text/html; charset=UTF-8">
  <script type="text/javascript" src="js/my.js"></script>
  <script type="text/javascript">
    function check name(){
      var xhr = getXhr();
      xhr.onreadystatechange=function(){
        if(xhr.readyState==4 && xhr.status==200){
          document.getElementById("name_msg")
            .innerHTML = xhr.responseText;
        }
      };
      xhr.setRequestHeader('content-type',
        'application/x-www-form-urlencoded');
      xhr.open("post","check_name.do",true);
      document.getElementById("name_msg")
        .innerHTML="正在检查。。";
      var uname = document.getElementById("uname");
      xhr.send("uname="+uname.value);
    }
  </script>
</head>
<body>
  <!-- 验证注册信息中的用户名是否可用 -->
  <form action="" method="post">
    <fieldset>
      <legend>注册信息</legend>
      用户名:<input name="uname" id="uname"/>
      <input type="button" value="检查一下吧"
        onclick="check_name()" />
      <span id="name_msg" style="color:red;"></span>
      <br/><br/>
      <input type="submit" value="注册"/>
    </fieldset>
  </form>
</body>
</html>
```

ActionServlet.java 文件代码如下：

```
package web;
```

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ActionServlet extends HttpServlet {

    public void service(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        //获取请求资源路径
        String uri = request.getRequestURI();
        String path = uri.substring(
            uri.lastIndexOf("/"),
            uri.lastIndexOf("."));
        if(path.equals("/get_text")){//get 请求
            out.println("来自星星的你");
        }else if(path.equals("/post_text")){//post 请求
            String name = request.getParameter("uname");
            System.out.println(name);
            out.println("又来了一次的" + name);
        }else if(path.equals("/check_name")){//检查用户名
            String name = request.getParameter("uname");
            //模拟网络延迟的操作
            if(1==1){
                try {
                    Thread.sleep(6000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
            System.out.println(name);
            if("Luffy".equals(name)){
                out.println("该用户名不可用");
            }else{
                out.println("可以使用!");
            }
        }
        out.close();
    }
}
```

5. POST 请求时的乱码处理

- 问题

在上例中，如果填写的用户名为中文信息时，就会出现如图-22 所示的结果。在控制台打印服务器端获取的数据为乱码。但火狐浏览器提交的中文可以被服务器端正确识别。原因是火狐会告诉服务器端以什么方式解码。而 IE 浏览器和 Chrome 浏览器提交的数据，到了服务器端都是以默认解码方式 ISO-8859-1 来解析的，而提交时浏览器依据 meta 标记指定的 UTF-8 的方式进行的编码，编码解码不一致就会导致乱码出现。如何解决？

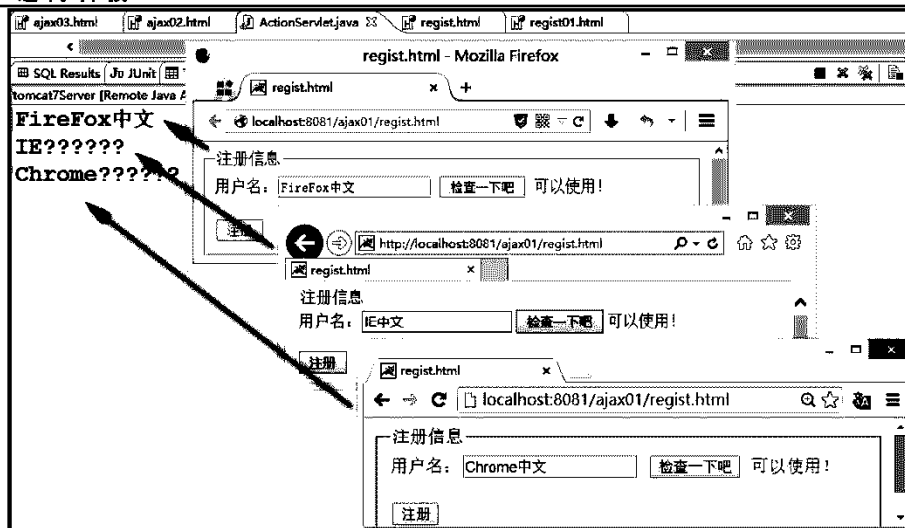


图 - 22

• 方案

在获取表单提交的数据之前，设置服务器端的解码方式为 UTF-8 即可。

• 步骤

步骤一：修改 ActionServlet 类

在 service 方法中增加一行代码即可：

```
response.setContentType("text/html;charset=UTF-8");
request.setCharacterEncoding("UTF-8");
PrintWriter out = response.getWriter();
```

图 - 23

步骤二：重新运行 regist.html 页面

重新运行程序，分别使用 Chrome、IE、Firefox 浏览器来访问 regist.html 页面，测试中文情况，结果如图-24 所示，可以正确得到表单以 POST 方式提交的中文字符了。

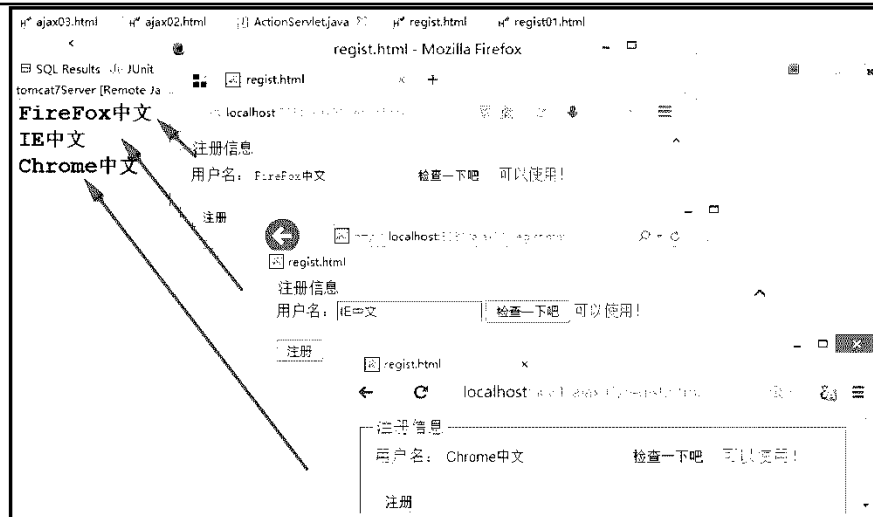


图 - 24

• 完整代码

regist.html 文件代码同上一案例。

ActionServlet.java 文件代码如下：

```
package web;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ActionServlet extends HttpServlet {

    public void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");
        PrintWriter out = response.getWriter();
        //获取请求资源路径
        String uri = request.getRequestURI();
        String path = uri.substring(
            uri.lastIndexOf("/"),
            uri.lastIndexOf("."));
        if(path.equals("/get_text")){//get 请求
            out.println("来自星星的你");
        }else if(path.equals("/post_text")){//post 请求
            String name = request.getParameter("uname");
            System.out.println(name);
            out.println("又来了一次的" + name);
        }
        else if(path.equals("/check name")){//检查用户名
            String name = request.getParameter("uname");
            //模拟一个耗时的操作
            if(1==1){
                try {
                    Thread.sleep(6000);
                } catch (InterruptedException e) {

```



```

        e.printStackTrace();
    }
}
System.out.println(name);
if("Luffy".equals(name)){
    out.println("该用户名不可用");
}else{
    out.println("可以使用!");
}
}
out.close();
}
}

```

6. GET 请求时的乱码处理

• 问题

在 regist.html 页面中，如果将表单的提交方式更换为 get，并且服务器端依然保留 UTF-8 的解码代码，运行程序时，依然会有如图-25 所示的乱码问题。如何解决？

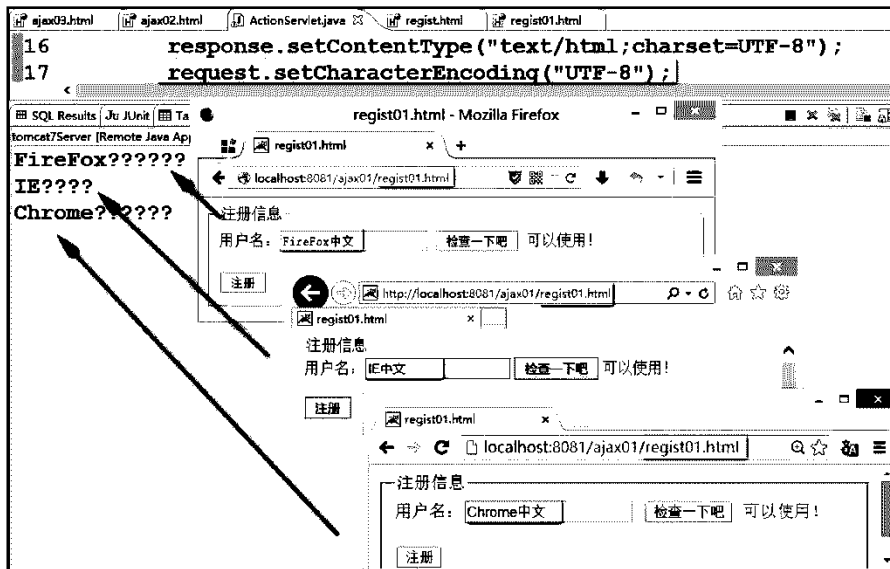


图 - 25

• 方案

产生乱码的原因是，以 GET 方式提交的数据存在于提交的消息头中 URL 中，HTTP 协议对于 URI 的编码方式为 UTF-8，但是 URI 后面的查询字符串的编码方式却是 ISO-8859-1。到了服务器端，容器针对 URI 的解码方式是 ISO-8859-1，依然解析不了中文，所以，会出现乱码。为了能够支持地址中查询字符串的中文格式，需要在浏览器端和服务器端同时进行支持 UTF-8 格式编码、解码的设置。

• 步骤

步骤一：修改 tomcat 的默认解码设置

修改 tomcat 安装路径下 conf/server.xml 文件。找到 Connector 节点后，增加 URIEncoding= "UTF-8 "这个属性即可。如图-26 所示：

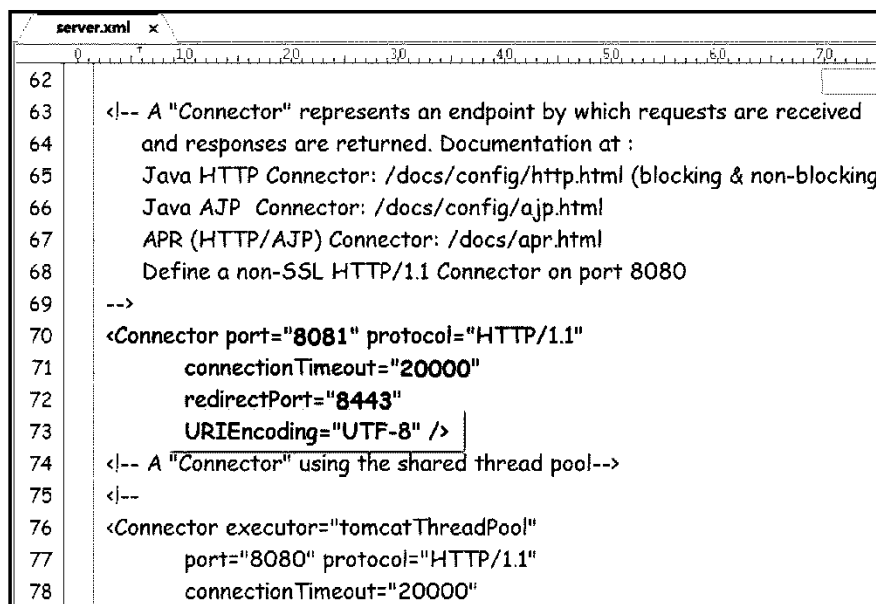


图 - 26

步骤二：新建 regist01.html 页面

新建 regist01.html 文件，表单内容参考 regist.html，Ajax 的请求修改为 GET 方式，为了让 URL 整体都使用 UTF-8 的方式进行编码，需要使用 JavaScript 语言中的 encodeURIComponent () 方法，所以在 open 方法创建请求时，第二个 URI 参数要使用 encodeURIComponent 方法进行编码。如图-27 所示：

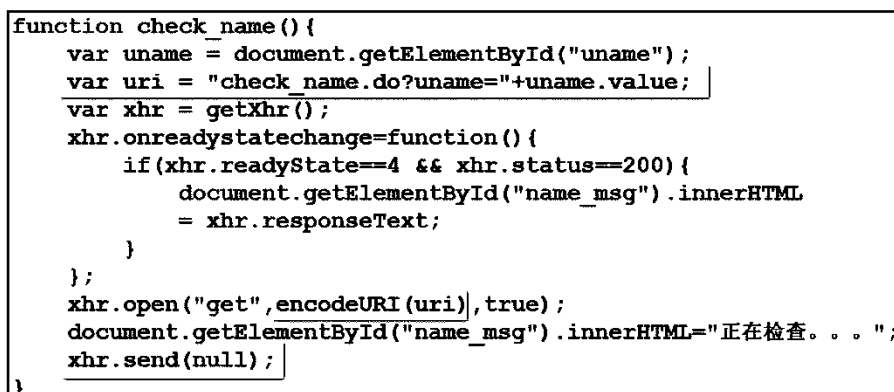


图 - 27

步骤三：ActionServlet 保持不变

步骤四：运行，查看结果

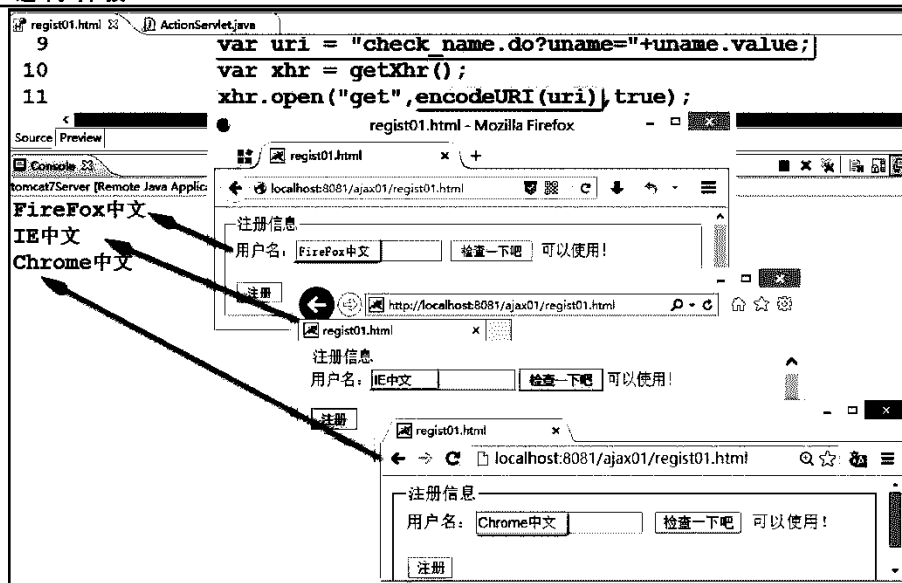


图 - 28

• 完整代码

regist01.html 文件代码如下：

```
<html>
<head>
<title>regist01.html</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script type="text/javascript" src="js/my.js"></script>
<script type="text/javascript">
function check_name(){
    var uname = document.getElementById("uname");
    var uri = "check_name.do?uname="+uname.value;
    var xhr = getXHR();
    xhr.onreadystatechange=function(){
        if(xhr.readyState==4 && xhr.status==200){
            document.getElementById("name_msg").innerHTML
            = xhr.responseText;
        }
    };
    xhr.open("get",encodeURIComponent(uri),true);
    document.getElementById("name_msg").innerHTML="正在检查...";
    xhr.send(null);
}
</script>
</head>
<body>
<!-- GET方式提交中文用户名进行验证 -->
<form action="" method="post">
<fieldset>
<legend>注册信息</legend>
用户名: <input name="uname" id="uname"/>
<input type="button" value="检查一下吧" onclick="check_name()"/>
<span id="name_msg" style="color:red;"></span>
<br/><br/>
<input type="submit" value="注册"/>
</fieldset>
</form>
</body>
</html>
```

ActionServlet.java 文件代码如下：

```
package web;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ActionServlet extends HttpServlet {

    public void service(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");
        PrintWriter out = response.getWriter();
        //获取请求资源路径
        String uri = request.getRequestURI();
        String path = uri.substring(
            uri.lastIndexOf("/"),
            uri.lastIndexOf("."));
        if(path.equals("/get_text")){//get 请求
            out.println("来自星星的你");
        }else if(path.equals("/post_text")){//post 请求
            String name = request.getParameter("uname");
            System.out.println(name);
            out.println("又来了一次的" + name);
        }else if(path.equals("/check_name")){//检查用户名
            String name = request.getParameter("uname");
            //模拟一个耗时的操作
            if(1==1){
                try {
                    Thread.sleep(6000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
            System.out.println(name);
            if("Luffy".equals(name)){
                out.println("该用户名不可用");
            }else{
                out.println("可以使用!");
            }
        }
        out.close();
    }
}
```

7. 使用 Ajax 实现级联的下拉列表

- 问题

实现市区的级联下拉列表。即，选择城市后，动态加载该城市包含的区域信息。

- **方案**

下拉列表发生 onchange 事件时，将选中的城市对应的 value 发送给服务器端，服务器端收到这个城市拼音 缩写后，进行比对，返回区域的组合信息。客户端收到信息后分别按照 “；”和“，”进行分割，然后构建 option 对象，添加到区域对应的下拉列表中。

- **步骤**

步骤一：修改 my.js 文件

修改 my.js 文件 增加两个便于定位 html 页面中元素的方法 ,以及便于读取元素 value 属性的方法。如图 - 29 所示：

```
function $(id) {  
    return document.getElementById(id);  
}  
  
function $F(id) {  
    return $(id).value;  
}
```

图 - 29

步骤二：新建 select.html 文件

编写页面中的下拉列表。并且在 onchange 事件时，将选中项的 value 属性作为参数传给 getCity 方法。如图-30 所示：

```
<body>  
    <!-- Ajax实现级联下拉列表 -->  
    <select id="s1" style="width:120px;"  
        onchange="getCity(this.value);">  
        <option value="bj">北京</option>  
        <option value="sh">上海</option>  
        <option value="gz">广州</option>  
    </select>  
    <select id="s2" style="width:120px;">  
    </select>  
</body>
```

图 - 30

步骤三：编写 Ajax 代码

```
function getCity(v1) {
    var xhr = getXhr();
    xhr.onreadystatechange = function() {
        if (xhr.readyState == 4 && xhr.status==200) {
            var txt = xhr.responseText;
            //静安,ja;黄浦,hp;浦东新,pdx
            var str = txt.split(';');
            //先清空s2
            $('#s2').innerHTML = '';
            for (i = 0; i < str.length; i++) {
                var str1 = str[i].split(',');
                /*构造一个Option对象 */
                var op = new Option(str1[0], str1[1]);
                /* options是select的一个属性,
                其值是一个数组。数组中的元素是Option对象。
                */
                $('#s2').options[i] = op;
            }
        }
    };
    xhr.open('get', 'getCity.do?name=' + v1, true);
    xhr.send(null);
}
```

图 - 31

步骤四：修改 ActionServlet

添加对 getCity.do 的请求处理：

```
}else if(path.equals("/getCity")){//城市列表联动
    String name = request.getParameter("name");
    if("bj".equals(name)){
        out.println("朝阳,cy;东城,dc");
    }else if("sh".equals(name)){
        out.println("静安,ja;黄浦,hp;浦东新,pdx");
    }else{
        out.println("白云,by;番禺,py");
    }
}
```

图 - 32

步骤五：访问，查看运行结果

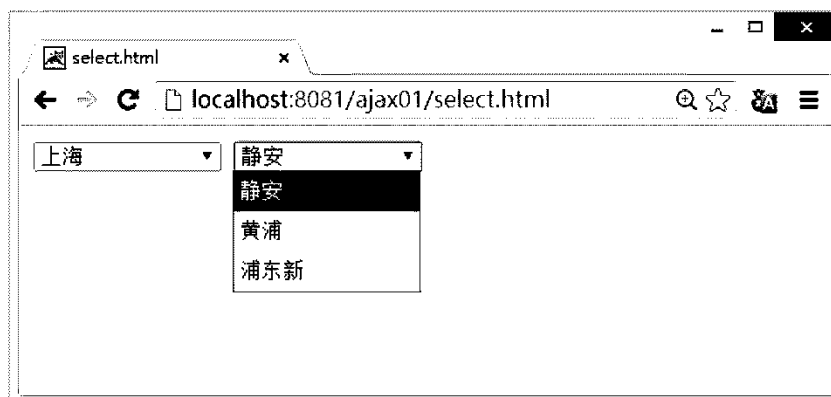


图- 33



图- 34

• 完整代码

js/my.js 文件代码如下：

```
/*获取 Ajax 对象*/
function getXHR(){
    var xhr = null;
    if(window.XMLHttpRequest){
        xhr = new XMLHttpRequest();
    }else{
        xhr = new ActiveXObject("Microsoft.XMLHttp");
    }
    return xhr;
}

function $(id){
    return document.getElementById(id);
}

function $F(id){
    return $(id).value;
}
```

select.html 文件代码如下：

```
<html>
<head>
<title>select.html</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script type="text/javascript" src="js/my.js"></script>
<script type="text/javascript">
    function getCity(v1) {
        var xhr = getXHR();
        xhr.onreadystatechange = function() {
            if (xhr.readyState == 4 && xhr.status==200) {
                var txt = xhr.responseText;
                //静安,ja;黄浦,hp;浦东新,pdx
                var strs = txt.split(';');
                //先清空 s2
                $('s2').innerHTML = '';
                for (i = 0; i < strs.length; i++) {
                    var str1s = strs[i].split(',');
                    /*构造一个 Option 对象 */
                    var op = new Option(str1s[0], str1s[1]);
                    /* options 是 select 的一个属性,
```

其值是一个数组。数组中的元素是 Option 对象。

```

        */
        $('s2').options[i] = op;
    }
}
};
xhr.open('get', 'getCity.do?name=' + v1, true);
xhr.send(null);
}
</script>
</head>
<body>
<!-- Ajax 实现级联下拉列表 -->
<select id="s1" style="width:120px;"
        onchange="getCity(this.value);">
    <option value="bj">北京</option>
    <option value="sh">上海</option>
    <option value="gz">广州</option>
</select>
<select id="s2" style="width:120px;">
</select>
</body>
</html>

```

ActionServlet.java 文件代码如下：

```

package web;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ActionServlet extends HttpServlet {

    public void service(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");
        PrintWriter out = response.getWriter();
        //获取请求资源路径
        String uri = request.getRequestURI();
        String path = uri.substring(
            uri.lastIndexOf("/"),
            uri.lastIndexOf("."));
        if(path.equals("/get text")){//get 请求
            out.println("来自星星的你");
        }else if(path.equals("/post text")){//post 请求
            String name = request.getParameter("uname");
            System.out.println(name);
            out.println("又来了一次的" + name);
        }
        else if(path.equals("/check_name")){//检查用户名
            String name = request.getParameter("uname");
            //模拟一个耗时的操作
            if(1==1){
                try {
                    Thread.sleep(6000);

```



```
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    System.out.println(name);
    if("Luffy".equals(name)){
        out.println("该用户名不可用");
    }else{
        out.println("可以使用! ");
    }
} else if(path.equals("/getCity")){//城市列表联动
    String name = request.getParameter("name");
    if("bj".equals(name)){
        out.println("朝阳,cy;东城,dc");
    }else if("sh".equals(name)){
        out.println("静安,ja;黄浦,hp;浦东新,pdx");
    }else{
        out.println("白云,by;番禺,py");
    }
}
}
out.close();
}
}
```

web.xml 文件代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
    <display-name></display-name>
    <servlet>
        <servlet-name>ActionServlet</servlet-name>
        <servlet-class>web.ActionServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>ActionServlet</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>
</web-app>
```

课后作业

1. 简述对 Ajax 的理解

2. 下列说法正确的是（ ）

- A. Ajax 的核心对象是 XMLHttpRequest 对象
- B. Ajax 的 XMLHttpRequest 对象的 status 属性代表响应的状态码
- C. 编写 Ajax 程序时，无需对浏览器进行判断即可创建 XMLHttpRequest 对象
- D. 编写 Ajax 程序时，需要为 readyStatechange 事件添加回调处理

3. 简述使用 AJAX 进行编程的步骤

4. 使用 Ajax 检验注册信息中的中文昵称是否存在

检查如图-1 所示的昵称是否存在。



图 - 1

如果昵称存在，界面显示效果如图-2 所示：

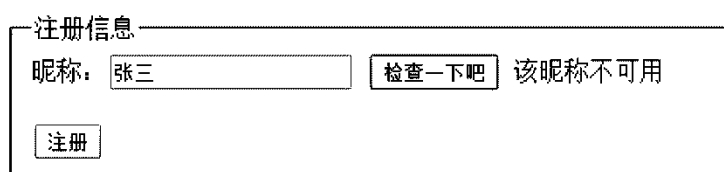


图 - 2

如果昵称不存在，界面显示效果如图-3 所示：

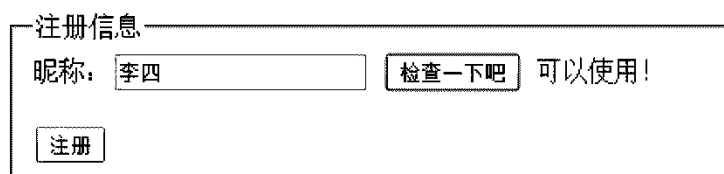


图 - 3

AJAX 和 jQuery

Unit02

知识体系.....Page 37

JSON	JSON 简介	什么是 JSON
		与平台无关的数据交换
		轻量级的解决方案
	JSON 语法	JSON 的结构
		使用 JSON 表示一个对象
		使用 JSON 表示一个数组
使用 JSON 实现数据交换	数据交换	数据交换原理
		JSON 字符串-->JavaScript 对象
		Java 对象转换前的准备
		Java 对象转换成 JSON
	缓存问题	什么是缓存问题
		如何解决缓存问题
	同步问题	什么是同步问题
		如何发送同步请求

经典案例.....Page 43

访问 JavaScript 对象的属性	使用 JSON 表示一个对象
访问 Javascript 对象数组	使用 JSON 表示一个数组
JSON 字符串转换成 JavaScript 对象	JSON 字符串-->JavaScript 对象
Java 对象转换成 JSON 字符串	Java 对象转换前的准备
使用 JSON 完成级联下拉列表	Java 对象转换成 JSON
热卖商品动态显示	如何解决缓存问题

课后作业.....Page 67

1. JSON

1.1. JSON 简介

1.1.1. 【JSON 简介】什么是 JSON

Tarena
达内科技

什么是JSON

- JSON (JavaScript Object Notation) 是一种轻量级的数据交换格式。
- 易于人阅读和编写，同时也易于机器解析和生成。
- JSON完全独立于语言之外,但语法上借鉴了JavaScript

+

1.1.2. 【JSON 简介】与平台无关的数据交换

Tarena
达内科技

与平台无关的数据交换

+

1.1.3. 【JSON 简介】轻量级的解决方案

Tarena
达内科技

轻量级的解决方案

轻量级：相对于XML，JSON解析速度更快，文档更小。

```
<emp>
  <name>Sally</name>
  <city>北京</city>
  <age>25</age>
</emp>
```

```
{ "name" : "Sally", "city" : "北京", "age" : 25 }
```

+

1.2. JSON 语法

1.2.1. 【JSON 语法】JSON 的结构

Technology
Tarena
达内科技

JSON的结构

- JSON主要分两种结构：
 - “名称/值”对的集合。不同的语言理解为对象、记录、结构、字典、哈希表等
 - 值的有序列表。大部分语言中理解为数组

Technology
Tarena
达内科技

1.2.2. 【JSON 语法】使用 JSON 表示一个对象

Technology
Tarena
达内科技

使用JSON表示一个对象

- { 属性名：属性值, 属性名：属性值 ... }
- 注意：
 - 属性值可以是 string, number, boolean (true, false), null, object.
 - 属性名必须使用双引号引起来
 - 属性值如果是字符串, 必须使用双引号括起来

Technology
Tarena
达内科技

1.2.3. 【JSON 语法】使用 JSON 表示一个数组

Technology
Tarena
达内科技

使用JSON表示一个数组

- JSON表示数组的语法是：[value, value, value]
- Value可以是简单数据类型, 也可以是object、数组类型
- 例如：

数组

```
[
  { "name" : "Jerry" , "age" : 22 },
  { "name" : "Tom" , "age" : 32 }
];
```

对象

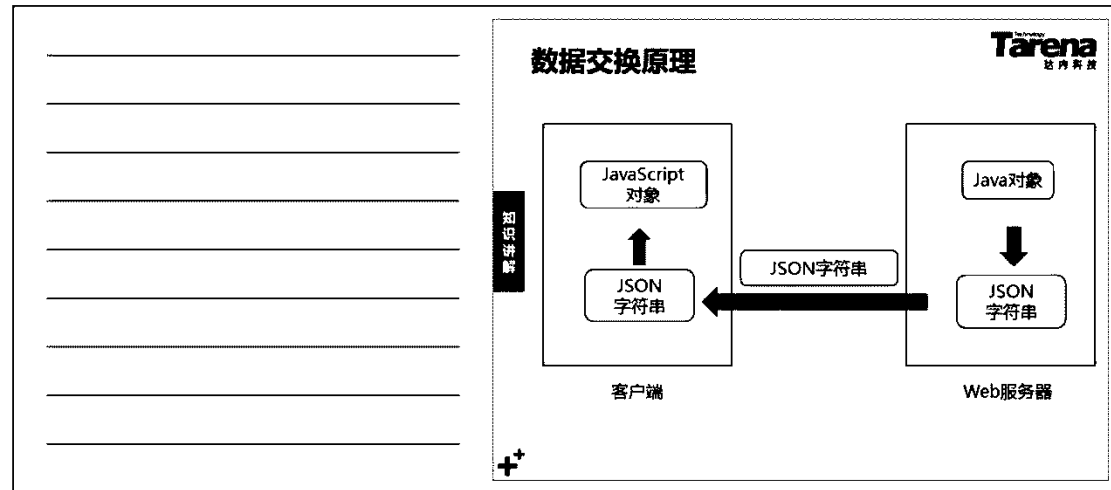
```
{
  "name" : "Jerry" ,
  "hobby" : [ "sing" , "dance" , "eat" ]
}
```

Technology
Tarena
达内科技

2. 使用 JSON 实现数据交换

2.1. 数据交换

2.1.1. 【数据交换】数据交换原理



2.1.2. 【数据交换】JSON 字符串-->JavaScript 对象

Tarena
达内科技

JSON字符串→JavaScript对象

- 使用JavaScript语言的原生函数：eval ()
- 但该方法有风险，使用需谨慎

```
var str = '{ "name": "张三", "age": 24 }';
var obj = eval( "(" + str + ")" );
alert ( obj.name );
```

++

Tarena
达内科技

JSON字符串→JavaScript对象(续1)

- 使用原生对象JSON提供的parse()方法
- 如果该对象无效，说明浏览器版本太低

```
var str = '{ "name": "王小贱", "age": 24 }';
var obj = JSON.parse ( str );
alert ( obj.name );
```

++

JSON字符串→JavaScript对象(续2)

- 使用JSON.js 文件为字符串增加的方法

```
var str = '{ "name": "王小贱", "age": 24 }';
var obj = str.parseJSON();
alert ( obj.name );
```

++

2.1.3. 【数据交换】Java 对象转换前的准备

Java对象转换前的准备

- 引入必要的jar文件
 - commons-beanutils.jar
 - commons-collections.jar
 - common-lang.jar
 - common-logging.jar
 - ezmorph.jar
 - json-lib.jar

++

2.1.4. 【数据交换】Java 对象转换成 JSON

Java对象转换成JSON


- 利用JSON官方提供的API : JSONObject , JSONArray 提供的方法。

```
Employee emp = new Employee( "1", "王小贱", "男" );
JSONObject obj = JSONObject.fromObject(emp);
String jsonStr = obj.toString();
```

++


2.2. 缓存问题


2.2.1. 【缓存问题】什么是缓存问题



什么是缓存问题


- IE浏览器提供的AJAX对象，在发送GET请求时，会先查看是否访问过该地址，如果该地址已经访问过，那么浏览器不再发送请求。






什么是缓存问题（续）

- 各浏览器对于同一地址的处理
 - Chrome 继续发请求
 - Firefox 继续发请求
 - IE浏览器 不再发请求



2.2.2. 【缓存问题】如何解决缓存问题




如何解决缓存问题

- 方式1：在请求地址后面添加一个随机数。

```
xhr.open( 'get ', 'getNumber.do?' + Math.random( ), true);
```

- 方式2：发送post请求



2.3. 同步问题

2.3.1. 【同步问题】什么是同步问题

什么是同步问题

- 发送同步请求时，浏览器要等待服务器的响应到达之后才能继续在页面中操作
- 当Ajax对象向服务器发送同步请求时，浏览器会锁定当前页面。

+

2.3.2. 【同步问题】如何发送同步请求

如何发送同步请求

- 调用异步对象的open方法时第三个参数传false
- 如：

```
xhr.open( 'get' , 'check...' , false );
```

+

经典案例

1. 访问 JavaScript 对象的属性

- 问题

访问 JavaScript 对象的属性。

- 方案

使用初始化的方式创建 JavaScript 对象，并访问对象的属性。

- 步骤

步骤一：新建 json01.html 文件

添加链接的单击动作，调用方法。

```
function f1(){
    var obj = {'name':'Luffy','age':17};
    alert(obj.name + " " + obj.age);
}
```

图 -1

步骤二：运行查看结果



图 - 2

步骤三：添加复杂对象

```
function f2(){
    var obj = {'name':'Luffy',
               'address':{'city':'Beijing',
                           'street':'dzs',
                           'room':17
                          }};
    alert(obj.name + " " + obj.address.city);
}
```

图 - 3

步骤四：运行查看结果



图 - 4

• 完整代码

json01.html 文件代码如下：

```
<html>
<head>
  <title>json01.html</title>
  <meta http-equiv="content-type"
    content="text/html; charset=UTF-8">
  <script type="text/javascript">
    function f1(){
      var obj = {'name':'Luffy','age':17};
      alert(obj.name + " " + obj.age);
    }
    function f2(){
      var obj = {'name':'Luffy',
        'address':{
          'city':'Beijing',
          'street':'dzs',
          'room':17
        }
      };
      alert(obj.name + " " + obj.address.city);
    }
  </script>
</head>
<body>
  <!-- 创建 JavaScript 对象并查看属性 -->
  <a href="javascript:;" onclick="f2();" >查看对象属性</a>
</body>
</html>
```

2. 访问 Javascript 对象数组

• 问题

访问 JavaScript 对象数组中的元素。

• 方案

遵循数组的格式要求，使用方括号开头和结尾。

• 步骤

步骤一：新建 json02.html 文件

在文件中添加链接进行方法调用的测试。添加第一个方法 f3()，如图-5 所示：

```
function f3(){
    var arr = [ {'name':'Luffy','age':17},
                {'name':'Zoro','age':19}];
    alert(arr[1].name);
}
```

图 - 5

步骤二：运行查看结果



图 - 6

步骤三：添加下一个方法 f4()

```
function f4(){
    var obj = {'name':'Luffy',
               'friends':[{'name':'Zoro','age':19},
                           {'name':'Nami','age':18}]
    };
    alert(obj.name + " " + obj.friends[1].name);
}
```

图 - 7

步骤四：运行查看结果

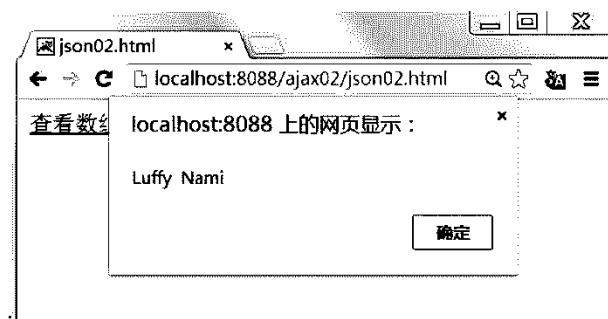


图 - 8

• 完整代码

json02.html 文件代码如下：

```
<html>
<head>
  <title>json02.html</title>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <script type="text/javascript">
    function f3(){
      var arr = [ {'name':'Luffy','age':17},
                  {'name':'Zoro','age':19}];
      alert(arr[1].name);
    }
    function f4(){
      var obj = {'name':'Luffy',
                 'friends':[{'name':'Zoro','age':19},
                             {'name':'Nami','age':18}]
                };
      alert(obj.name + " " + obj.friends[1].name);
    }
  </script>
</head>
<body>
  <!-- 访问 JavaScript 对象数组 -->
  <a href="javascript:;" onclick="f4();">查看数组中的对象属性</a>
</body>
</html>
```

3. JSON 字符串转换成 JavaScript 对象

- 问题

将符合 JSON 格式要求的字符串，转换为 JavaScript 对象。

- 方案

借助于原生的 eval 方法或者原生对象 JSON.parse(str)方法。

- 步骤

步骤一：新建 json03.html 文件

在文件中添加脚本代码，f5（）方法为字符串到对象的转变过程。

JSON 这个原生对象提供了字符串和对象之间转换的方法，如果浏览器无法得到想要的输出结果，是因为版本太低不支持这个原生对象。

```
/*JSON字符串转JSON对象*/
function f5(){
    var str = '{"name":"Luffy","age":17}';

    //第1种方式(不需要任何js文件)
    //var obj = eval("(" + str + ")");

    //第2种方式(不需要json.js文件)
    //var obj = JSON.parse(str);

    //第3种方式(需要json.js文件)
    var obj = str.parseJSON();

    alert(obj.name);
}
```

图 - 9

步骤二：运行查看结果



图 - 10

步骤三：添加方法 f6(),实现数组的转换

```
/*JSON字符串转JSON数组*/
function f6(){
    var str = ' [{"name":"Luffy","age":17}, ' +
                ' {"name":"Zoro","age":19}]';

    //第1种方式(不需要任何js文件)
    //var arr = eval("(" + str + ")");

    //第2种方式(需要json.js文件)
    var arr = str.parseJSON();

    alert(arr[1].name);
}
```

图 - 11

步骤四：运行查看结果

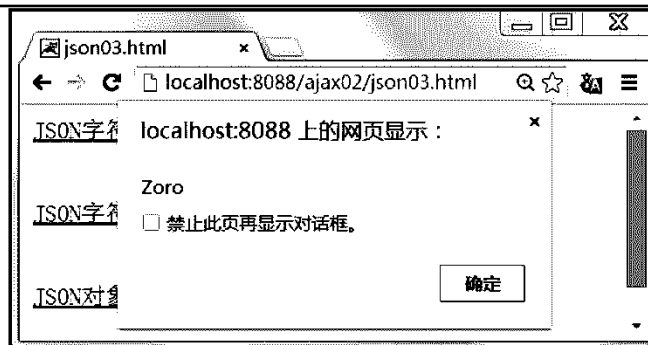


图 - 12

步骤五：添加 f7() 方法，实现对象到字符串的转换

```
/*JSON对象转JSON字符串*/
function f7(){
    var obj = {"name":"Luffy","age":17};
    //第1种方式(需要json.js文件)
    //var str = obj.toJSONString();
    //第2种方式(不需要任何js文件)
    var str = JSON.stringify(obj);
    alert(str);
}
```

图- 13

步骤六：运行查看结果



图 - 14

• 完整代码

```
<html>
<head>
<title>json03.html</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">

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

    /*JSON字符串转JSON对象*/
    function f5(){
        var str = '{"name":"Luffy","age":17}';
```

```
//第1种方式(不需要任何js文件)
//var obj = eval("(" + str + ")");

//第2种方式(不需要json.js文件)
//var obj = JSON.parse(str);

//第3种方式(需要json.js文件)
var obj = str.parseJSON();

alert(obj.name);
}

/*JSON 字符串转 JSON 数组*/
function f6(){
    var str = '[{"name":"Luffy","age":17}, ' +
        '{"name":"Zoro","age":19}]';

    //第1种方式
    //var arr = eval("(" + str + ")");

    //第2种方式
    var arr = str.parseJSON();

    alert(arr[1].name);
}

/*JSON 对象转 JSON 字符串*/
function f7(){
    var obj = {"name":"Luffy","age":17};

    //第1种方式
    //var str = obj.toJSONString();

    //第2种方式
    var str = JSON.stringify(obj);
    alert(str);
}
</script>
</head>
<body>
    <!-- 使用 JSON 表示数组 -->
    <a href="javascript:;" onclick="f5();">JSON 字符串 -->JSON 对象
</a><br/><br/><br/>
    <a href="javascript:;" onclick="f6();">JSON 字符串 -->JSON 数组
</a><br/><br/><br/>
    <a href="javascript:;" onclick="f7();">JSON 对象 -->JSON 字符串
</a><br/><br/><br/>
</body>
</html>
```

4. Java 对象转换成 JSON 字符串

• 问题

将 Java 对象转换成符合 JSON 格式的字符串，并测试。

• 方案

使用与 json-lib.jar 相关的 jar 文件完成类型的转换。

- 步骤

步骤一：新建实体类 Friend

```
1 package bean;
2
3 public class Friend {
4     private String name;
5     private int age;
6
7     public String getName() {
8         return name;
9     }
10
11    public void setName(String name) {
12        this.name = name;
13    }
14
15    public int getAge() {
16        return age;
17    }
18
19    public void setAge(int age) {
20        this.age = age;
21    }
22
23    public String toString() {
24        return this.name + "    " + this.age;
25    }
26 }
```

图 - 15

步骤二：新建 JSONTest 类

在该类中添加四个方法，分别用于测试 Java 对象转换为 JSON 字符串，Java 数组转换为 JSON 字符串，JSON 字符串转换为 Java 对象，JSON 字符串转换为 Java 数组。

步骤三：引入 jar 文件

在添加 jar 文件的时候，需要导入 6 个 jar 文件，缺一不可。如图-16 所示：



图 - 16

步骤四：添加具体的转换方法：Java 对象转换为 JSON 字符串

```
/**
 * Java对象转换为JSON字符串
 */
public static void test1() {
    Friend f = new Friend();
    f.setName("Zoro");
    f.setAge(19);
    JSONObject jsonObj = JSONObject.fromObject(f);
    String jsonStr = jsonObj.toString();
    System.out.println(jsonStr);
}
```

图 - 17

步骤五：运行结果

```
{ "age":19, "name": "Zoro" }
```

图 - 18

步骤六：Java 数组（集合）转换为 JSON 字符串

```
public static void test2() {
    List<Friend> fs = new ArrayList<Friend>();
    for (int i = 0; i < 3; i++) {
        Friend f = new Friend();
        f.setName("Zoro" + (i + 1));
        f.setAge(19 + i);
        fs.add(f);
    }
    JSONArray jsonArr = JSONArray.fromObject(fs);
    String jsonStr = jsonArr.toString();
    System.out.println(jsonStr);
}
```

图 - 19

运行结果：

```
[{"age":19, "name": "Zoro1"}, {"age":20, "name": "Zoro2"}, {"age":21, "name": "Zoro3"}]
```

图 - 20

步骤七：JSON 字符串转换为 Java 对象

```
/**
 * JSON字符串转换为Java对象
 */
public static void test3(){
    String jsonStr = "{\"name\":\"Luffy\",\"age\":17}";
    JSONObject jsonObj = JSONObject.fromObject(jsonStr);
    Friend friend = (Friend)JSONObject.
        toBean(jsonObj, Friend.class);
    System.out.println(friend);
}
```

图 - 21

运行结果：

Luffy	17
-------	----

图 - 22

步骤八：JSON 字符串转换为 Java 数组（集合）

```
public static void test4(){
    String jsonStr = "[{\"name\":\"Luffy\",\"age\":17},{\"name\":\"Zoro\",\"age\":19}]";
    JSONArray jsonArr = JSONArray.fromObject(jsonStr);
    List<Friend> friends = (List<Friend>)JSONArray.
        toCollection(jsonArr, Friend.class);
    for(Friend f : friends){
        System.out.println(f);
    }
}
```

图 - 23

运行结果：

Luffy	17
Zoro	19

图 - 24

• 完整代码

Friend.java 文件代码如下：

```
package bean;

public class Friend {
    private String name;
    private int age;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String toString() {
        return this.name + " " + this.age;
    }
}
```

JSONTest.java 文件代码如下：

```
package json;

import java.util.ArrayList;
import java.util.List;

import net.sf.json.JSONArray;
import net.sf.json.JSONObject;

import bean.Friend;

public class JSONTest {
    /**
     * Java 对象转换为 JSON 字符串
     */
    public static void test1() {
        Friend f = new Friend();
        f.setName("Zoro");
        f.setAge(19);
        JSONObject jsonObj = JSONObject.fromObject(f);
        String jsonStr = jsonObj.toString();
        System.out.println(jsonStr);
    }

    /**
     * Java 数组转换为 JSON 字符串
     */
    public static void test2() {
        List<Friend> fs = new ArrayList<Friend>();
        for (int i = 0; i < 3; i++) {
            Friend f = new Friend();
            f.setName("Zoro" + (i + 1));
            f.setAge(19 + i);
            fs.add(f);
        }
        JSONArray jsonArr = JSONArray.fromObject(fs);
        String jsonStr = jsonArr.toString();
        System.out.println(jsonStr);
    }

    /**
     * JSON 字符串转换为 Java 对象
     */
    public static void test3(){
        String jsonStr = "[{\"name\":\"Luffy\",\"age\":17}";
        JSONObject jsonObj = JSONObject.fromObject(jsonStr);
        Friend friend = (Friend)JSONObject.
            toBean(jsonObj, Friend.class);
        System.out.println(friend);
    }

    /**
     * JSON 字符串转换为 Java 数组
     */
    public static void test4(){
        String jsonStr = "[[{\"name\":\"Luffy\",\"age\":17},\" +
            \"[{\"name\":\"Zoro\",\"age\":19}]]";
        JSONArray jsonArr = JSONArray.fromObject(jsonStr);
        List<Friend> friends = (List<Friend>)JSONArray.
            toCollection(jsonArr, Friend.class);
        for(Friend f :friends){
            System.out.println(f);
        }
    }
}
```

```
public static void main(String[] args) {  
    //test1();  
    //test2();  
    //test3();  
    test4();  
}  
}
```

5. 使用 JSON 完成级联下拉列表

- 问题

结合异步请求，实现城市的级联下拉列表。

- 方案

分别编写客户端脚本和服务端处理程序。服务器端要实现将 Java 对象转换为传输的 JSON 字符串。客户端在收到这个字符串以后进行转换，变成 JavaScript 对象，使用对象的各个属性构造下拉框的 Option 选项后添加到 select 下面。

- 步骤

步骤一：新建实体类 City

```
package bean;  
  
public class City {  
    private String cityName;  
    private String cityValue;  
  
    public City() {  
        super();  
    }  
    public City(String cityName, String cityValue) {  
        super();  
        this.cityName = cityName;  
        this.cityValue = cityValue;  
    }  
    public String getCityName() {  
        return cityName;  
    }  
    public void setCityName(String cityName) {  
        this.cityName = cityName;  
    }  
    public String getCityValue() {  
        return cityValue;  
    }  
    public void setCityValue(String cityValue) {  
        this.cityValue = cityValue;  
    }  
}
```

图 - 25

步骤二：新建 ActionServlet

```
String uri = request.getRequestURI();
String action = uri.substring(uri.lastIndexOf("/")
,uri.lastIndexOf("."));
if (action.equals("/city")) {
    String name = request.getParameter("name");
    if (name.equals("bj")) {
        City c1 = new City("海淀", "hd");
        City c2 = new City("东城", "dc");
        City c3 = new City("西城", "xc");
        List<City> cs = new ArrayList<City>();
        cs.add(c1);
        cs.add(c2);
        cs.add(c3);
        JSONArray obj = JSONArray.fromObject(cs);
        String str = obj.toString();
        out.println(str);
    } else if (name.equals("sh")) {
        City c1 = new City("徐汇", "xh");
        City c2 = new City("静安", "ja");
        City c3 = new City("黄浦", "hp");
        List<City> cs = new ArrayList<City>();
        cs.add(c1);
        cs.add(c2);
        cs.add(c3);
        JSONArray obj = JSONArray.fromObject(cs);
        String str = obj.toString();
        out.println(str);
    }
}
```

图 - 26

步骤三：新建 city.html 文件

```
<body>
    <select id="s1" style="width:120px;"
    onchange="change(this.value);">
        <option value="sh">上海</option>
        <option value="bj">北京</option>
    </select>
    <select id="s2" style="width:120px;">
    </select>
</body>
```

图 - 27

```
function change(v1){
    var xhr = getXmlHttpRequest();
    xhr.open('post','city.do',true);
    xhr.setRequestHeader("Content-Type",
        "application/x-www-form-urlencoded");
    xhr.onreadystatechange=function(){
        if(xhr.readyState == 4){
            var txt = xhr.responseText;
            var citys = txt.parseJSON();
            document.getElementById('s2').innerHTML = '';
            for(i=0;i<citys.length;i++){
                var op =
                    new Option(citys[i].cityName,
                        citys[i].cityValue);
                document.getElementById('s2').options[i] = op;
            }
        }
    };
    xhr.send('name=' + v1);
}
```

图 - 28

步骤四：运行查看结果



图 - 29

• 完整代码

City.java 文件代码如下：

```
package bean;

public class City {
    private String cityName;
    private String cityValue;

    public City() {
        super();
    }
    public City(String cityName, String cityValue) {
        super();
        this.cityName = cityName;
        this.cityValue = cityValue;
    }
    public String getCityName() {
        return cityName;
    }
}
```

```

    }
    public void setCityName(String cityName) {
        this.cityName = cityName;
    }
    public String getCityValue() {
        return cityValue;
    }
    public void setCityValue(String cityValue) {
        this.cityValue = cityValue;
    }
}

```

ActionServlet.java 文件代码如下：

```

package web;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import dao.SaleDAO;
import json.DateProcessor;

import net.sf.json.JSONArray;
import net.sf.json.JSONObject;
import net.sf.json.JsonConfig;

import bean.City;
import bean.Sale;
import bean.User;

public class ActionServlet extends HttpServlet {

    public void service(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        String uri = request.getRequestURI();
        String action = uri.substring(uri.lastIndexOf("/")
,uri.lastIndexOf("."));
        if (action.equals("/city")) {
            String name = request.getParameter("name");
            if (name.equals("bj")) {
                City c1 = new City("海淀", "hd");
                City c2 = new City("东城", "dc");
                City c3 = new City("西城", "xc");
                List<City> cs = new ArrayList<City>();
                cs.add(c1);
                cs.add(c2);
                cs.add(c3);
                JSONArray obj = JSONArray.fromObject(cs);
                String str = obj.toString();
                out.println(str);
            } else if (name.equals("sh")) {
                City c1 = new City("徐汇", "xh");

```



```

        City c2 = new City("静安", "ja");
        City c3 = new City("黄浦", "hp");
        List<City> cs = new ArrayList<City>();
        cs.add(c1);
        cs.add(c2);
        cs.add(c3);
        JSONArray obj = JSONArray.fromObject(cs);
        String str = obj.toString();
        out.println(str);
    }
}
out.close();
}
}

```

web.xml 文件代码如下：

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
    <display-name></display-name>
    <servlet>
        <servlet-name>ActionServlet</servlet-name>
        <servlet-class>web.ActionServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>ActionServlet</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>
</web-app>

```

city.html 文件代码如下：

```

<html>
<head>
<title></title>
<script type="text/javascript" src="js/json.js"></script>
<script type="text/javascript">
    function getXmlHttpRequest(){
        var xhr = null;
        if((typeof XMLHttpRequest) != 'undefined'){
            xhr = new XMLHttpRequest();
        }else {
            xhr = new ActiveXObject('Microsoft.XMLHttp');
        }
        return xhr;
    }

    function change(v1){
        var xhr = getXmlHttpRequest();
        xhr.open('post', 'city.do', true);
        xhr.setRequestHeader("Content-Type",
            "application/x-www-form-urlencoded");
        xhr.onreadystatechange=function(){
            if(xhr.readyState == 4){
                var txt = xhr.responseText;
                var citys = txt.parseJSON();
                document.getElementById('s2').innerHTML = '';
                for(i=0;i<citys.length;i++){

```

```

        var op =
        new Option(citys[i].cityName,
        citys[i].cityValue);
        document.getElementById('s2').options[i] = op;
    }
}
};
xhr.send('name=' + v1);
}
</script>
</head>
<body>
<select id="s1" style="width:120px;"
onchange="change(this.value);">
    <option value="sh">上海</option>
    <option value="bj">北京</option>
</select>
<select id="s2" style="width:120px;">
</select>
</body>
</html>

```

6. 热卖商品动态显示

- **问题**

每隔 5 秒钟，显示当前卖的最好的三件商品。

- **方案**

每 5 秒钟发送一次 Ajax 请求，将返回的 JSON 数组数据显示到页面的 div 中。

- **步骤**

步骤一：新建 Sale 类

```
package bean;

public class Sale {
    private int id ;
    private String prodName;
    private int qty ;

    public Sale() {}
    public Sale(int id, String prodName, int qty)
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getProdName() {
        return prodName;
    }
    public void setProdName(String prodName) {
        this.prodName = prodName;
    }
    public int getQty() {
        return qty;
    }
    public void setQty(int qty) {
        this.qty = qty;
    }
}
```

图 - 30

步骤二：新建 dao 包下面的 DBUtil 类和 SaleDAO 类

DBUtil.java 文件参考前面项目中的代码。

执行如下 sql 语句：

```
Connection: |
create table t_sale(
    id number(6) primary key,
    prodName varchar2(20),
    qty number(4))

create sequence sale_id_seq increment by 1 start with 1

select * from (select rownum r,a.* from
(select * from t_sale order by qty desc) a) c where c.r < 4
```

图 - 31

新建 SaleDAO.java 文件，编写用于查询销量前三的方法。

```
public class SaleDAO {
    public List<Sale> findAll() throws Exception{
        List<Sale> sales = new ArrayList<Sale>();
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
        try{
            conn = DBUtil.getConnection();
            stmt = conn.prepareStatement("select * from (select rownum r," +
                "a.* from (select * from t_sale order by qty desc) a) " +
                "c where c.r < 4");
            rs = stmt.executeQuery();
            while(rs.next()){
                Sale s = new Sale(
                    rs.getInt("id"),
                    rs.getString("prodname"),
                    rs.getInt("qty")
                );
                sales.add(s);
            }
        }catch(Exception e){
            e.printStackTrace();
            throw e;
        }finally{
            DBUtil.close(conn);
        }
        return sales;
    }
}
```

图- 32

步骤三：新建 ActionServlet 类

```
public class ActionServlet extends HttpServlet {

    public void service(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();
        String uri = request.getRequestURI();
        String action = uri.substring(uri.lastIndexOf("/")
            ,uri.lastIndexOf("."));
        if(action.equals("/sale")){
            SaleDAO dao = new SaleDAO();
            try {
                List<Sale> all = dao.findAll();
                JSONArray array = JSONArray.fromObject(all);
                out.println(array.toString());
            } catch (Exception e) {
                e.printStackTrace();
                throw new ServletException(e);
            }
        }
        out.close();
    }
}
```

图 - 33

步骤四：新建 sales.html 文件

```
function f2(){
    var xhr = getXmlHttpRequest();
    xhr.open('post','sale.do',true);
    xhr.setRequestHeader("Content-Type",
        "application/x-www-form-urlencoded");
    xhr.onreadystatechange=function(){
        if(xhr.readyState == 4){
            var txt = xhr.responseText;
            var sales = txt.evalJSON();
            var saleInfo = '当前销量最好的商品是:<br/>';
            for(i=0;i<sales.length;i++){
                saleInfo += '商品名称:'
                    + sales[i].prodName + ' 销量:' +
                    sales[i].qty + '<br/>';
            }
            $('d1').innerHTML = saleInfo;
        }
    };
    xhr.send(null);
}
```

图- 34

步骤五：运行查看结果

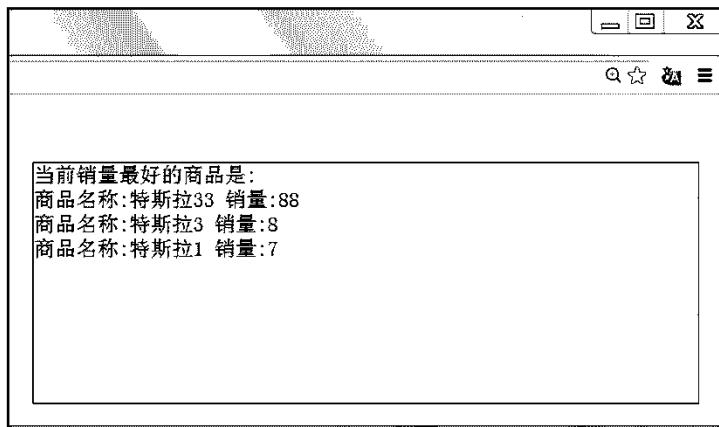


图 - 35

• 完整代码

Sale.java 文件代码：

```
package bean;

public class Sale {
    private int id ;
    private String prodName;
    private int qty ;

    public Sale() {
        super();
    }
    public Sale(int id, String prodName, int qty) {
        super();
        this.id = id;
    }
}
```

```

        this.prodName = prodName;
        this.qty = qty;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getProdName() {
        return prodName;
    }
    public void setProdName(String prodName) {
        this.prodName = prodName;
    }
    public int getQty() {
        return qty;
    }
    public void setQty(int qty) {
        this.qty = qty;
    }
}

```

DBUtil.java 文件代码:

```

package dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import org.apache.taglibs.standard.tag.common.core.OutSupport;

/**
 * JDBC 管理连接的工具类，可以获取连接和关闭连接
 */
public class DBUtil {
    /**
     * 获取连接对象
     */
    public static Connection getConnection()throws Exception{
        Connection conn = null;
        try {
            Class.forName("oracle.jdbc.OracleDriver");
            conn = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe","xxx", "xxx");
        } catch (Exception e) {
            e.printStackTrace();
            throw e;
        }
        return conn;
    }
    /**
     * 关闭连接对象
     */
    public static void close(Connection conn) throws Exception{
        if(conn!=null){
            try {
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
                throw e;
            }
        }
    }
}

```

```
}  
}
```

SaleDAO.java 文件代码：

```
package dao;  
  
import java.sql.Connection;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.util.ArrayList;  
import java.util.List;  
  
import bean.Sale;  
  
public class SaleDAO {  
    public List<Sale> findAll() throws Exception{  
        List<Sale> sales = new ArrayList<Sale>();  
        Connection conn = null;  
        PreparedStatement stmt = null;  
        ResultSet rs = null;  
        try{  
            conn = DBUtil.getConnection();  
            stmt = conn.prepareStatement("select * from (select rownum r," +  
                "a.* from (select * from t sale order by qty desc) a) " +  
                "c where c.r < 4");  
            rs = stmt.executeQuery();  
            while(rs.next()){  
                Sale s = new Sale(  
                    rs.getInt("id"),  
                    rs.getString("prodname"),  
                    rs.getInt("qty")  
                );  
                sales.add(s);  
            }  
        }catch(Exception e){  
            e.printStackTrace();  
            throw e;  
        }finally{  
            DBUtil.close(conn);  
        }  
        return sales;  
    }  
}
```

ActionServlet.java 文件代码：

```
package web;  
  
import java.io.IOException;  
import java.io.PrintWriter;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
  
import javax.servlet.ServletException;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
import dao.SaleDAO;  
  
import json.DateProcessor;
```

```
import net.sf.json.JSONArray;
import net.sf.json.JSONObject;
import net.sf.json.JsonConfig;

import bean.City;
import bean.Sale;
import bean.User;

public class ActionServlet extends HttpServlet {

    public void service(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        String uri = request.getRequestURI();
        String action = uri.substring(uri.lastIndexOf("/")
,uri.lastIndexOf("."));

        if(action.equals("/sale")){
            SaleDAO dao = new SaleDAO();
            try {
                List<Sale> all = dao.findAll();
                JSONArray array = JSONArray.fromObject(all);
                out.println(array.toString());
            } catch (Exception e) {
                e.printStackTrace();
                throw new ServletException(e);
            }
        }
        out.close();
    }
}
```

web.xml 文件代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app 2.5.xsd">
    <display-name></display-name>
    <servlet>
        <servlet-name>ActionServlet</servlet-name>
        <servlet-class>web.ActionServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>ActionServlet</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>
</web-app>
```

sales.html 文件代码：

```
<html>
<head>
<title>Insert title here</title>
<style>
    #d1{
        width:500px;
        height:180px;
```



```
background-color:#fff8dc;
border:1px solid red;
margin-left:350px;
margin-top:50px;
}
</style>
<script type="text/javascript" src="js/prototype-1.6.0.3.js"></script>
<script type="text/javascript">
function getXmlHttpRequest(){
    var xhr = null;
    if((typeof XMLHttpRequest)!='undefined'){
        xhr = new XMLHttpRequest();
    }else {
        xhr = new ActiveXObject('Microsoft.XMLHttp');
    }
    return xhr;
}
function f1(){
    setInterval(f2,5000);
}
function f2(){
    var xhr = getXmlHttpRequest();
    xhr.open('post','sale.do',true);
    xhr.setRequestHeader("Content-Type",
        "application/x-www-form-urlencoded");
    xhr.onreadystatechange=function(){
        if(xhr.readyState == 4){
            var txt = xhr.responseText;
            var sales = txt.evalJSON();
            var saleInfo = '当前销量最好的商品是:<br/>';
            for(i=0;i<sales.length;i++){
                saleInfo += '商品名称:'
                    + sales[i].prodName + ' 销量:' +
                    sales[i].qty + '<br/>';
            }
            $('d1').innerHTML = saleInfo;
        }
    };
    xhr.send(null);
}
</script>
</head>
<body onload="f1();">
<div id="d1"></div>
</body>
</html>
```

课后作业

1. 简述对 JSON 的理解
2. 编写程序，使用 JSON 实现三级联动的下拉列表
3. 下列说法正确的是（ ）
 - A. JSON 字符串转换为 JavaScript 对象时只能依靠 eval()方法
 - B. 因请求地址不变，导致有些浏览器认为不需要请求新的数据，而继续使用原有页面的过程叫做客户端缓存
 - C. AJAX 所发出的请求是同步请求
 - D. JSON 是一种轻量级的数据交换格式

AJAX 和 jQuery

Unit03

知识体系.....Page 70

jQuery	jQuery 介绍	什么是 jQuery
	jQuery 的编程步骤	浏览器如何解析 HTML 文档
		利用选择器定位节点
		调用方法操作节点
		jQuery 的编程步骤
	jQuery 对象与 DOM 对象之间的转换	什么是 jQuery 对象
		DOM 对象 -> jQuery 对象
		jQuery 对象 -> DOM 对象
jQuery 选择器	jQuery 选择器	什么是 jQuery 选择器
		选择器的种类
	基本选择器	#id
		.class
		element
		selector1,selector2,...selectorN
		*
	层次选择器	select1 空格 select2
		select1 > select2
		select1+select2
		select1~select2
	过滤选择器	基本过滤选择器
		内容过滤选择器
		可见性过滤选择器
		属性过滤选择器
		子元素过滤选择器
		表单对象属性过滤选择器
	表单选择器	表单选择器
jQuery 操作 DOM	jQuery 操作 DOM-查询	html ()
		text ()
		val ()
		attr ()

	jQuery 操作 DOM-创建、插入、删除	创建 DOM 节点
		插入 DOM 节点
		删除 DOM 节点
	jQuery 操作 DOM – 复制节点	复制 DOM 节点
	jQuery 操作 DOM – 样式操作	样式操作
	jQuery 操作 DOM – 遍历节点	遍历节点

经典案例.....Page 84


jQuery 简单案例	调动方法操作节点
	jQuery 的编程步骤
jQuery 对象与 DOM 对象的转换	DOM 对象->jQuery 对象
	jQuery 对象->DOM 对象
jQuery 选择器	#id
	.class
	element
	selector1 , selector2...
	*
	select1 空格 select2
	select1 > select2
	select+select
	select~select
	基本过滤选择器
	内容过滤选择器
	可见性过滤选择器
	属性过滤选择
	子元素过滤选择器
	表单对象属性过滤选择器
jQuery 操作 DOM	表单选择器
	html ()
	text ()
	val()
	attr()
	创建 DOM 节点
	插入 DOM 节点
	删除 DOM 节点
	复制 DOM 节点
	样式操作
	遍历节点

课后作业.....Page 125

1. jQuery

1.1. jQuery 介绍


1.1.1. 【jQuery 介绍】什么是 jQuery



什么是jQuery

- jQuery是一个优秀的JavaScript框架。一个轻量级的JavaScript库。
- 兼容CSS3，及各种浏览器
- 使用户更方便地处理HTML、Events、实现动画效果，并且方便地为网站提供AJAX交互
- 使用户的HTML页面保持代码和HTML内容分离
- 注：jQuery 2.x 开始不再支持 *Internet Explorer 6, 7, 8*

+




什么是jQuery（续）

- jQuery的核心理念是write less，do more
- 2006年1月发布

+

1.2. jQuery 的编程步骤

1.2.1. 【jQuery 的编程步骤】浏览器如何解析 HTML 文档

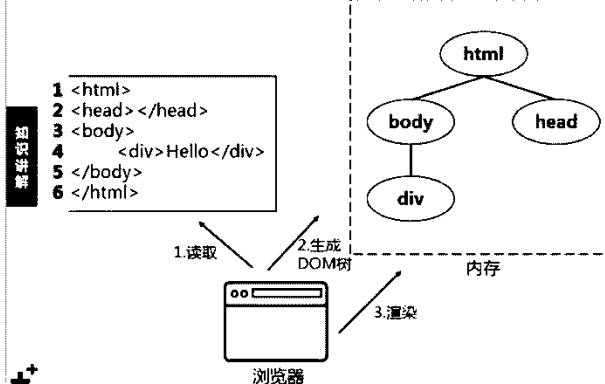


浏览器如何解析HTML文档

```

1 <html>
2 <head> </head>
3 <body>
4   <div>Hello</div>
5 </body>
6 </html>

```



1.2.2. 【jQuery 的编程步骤】利用选择器定位节点

Tarena
达内科技

利用选择器定位节点

代码片段

```
<html>
<head></head>
<body>
  <div id= "d1" >Hello</div>
</body>
</html>
```

选择器
定位

\$("div")

选择器
定位

\$("#d1")

+

1.2.3. 【jQuery 的编程步骤】调用方法操作节点

Tarena
达内科技

调用方法操作节点

代码片段

```
<html>
<head></head>
<body>
  <div id= "d1" >Hello</div>
</body>
</html>
```

操作
节点

\$("div")
.css("font-size" , "30px")

操作
节点

\$("#d1")
.css("font-size" , "30px")

+

1.2.4. 【jQuery 的编程步骤】jQuery 的编程步骤

Tarena
达内科技


jQuery的编程步骤

- ① 引入jQuery的js文件
- ② 使用选择器定位要操作的节点
- ③ 调用jQuery的方法进行操作

+


1.3. jQuery 对象与 DOM 对象之间的转换

1.3.1. 【jQuery 对象与 DOM 对象之间的转换】什么是 jQuery 对象




什么是jQuery对象

- jQuery为了解决浏览器的兼容问题而提供的一种统一封装后的对象描述
- jQuery提供的方法都是针对jQuery对象特有的，而且大部分方法的返回值类型也是jQuery对象，所以方法可以连缀调用 "jQuery对象.方法().方法().方法()..."
- 如：\$("div")和\$("#d1")都是jQuery对象



1.3.2. 【jQuery 对象与 DOM 对象之间的转换】DOM 对象 -> jQuery 对象




DOM对象 -> jQuery对象

- 实现方式：


\$(DOM对象)

如：

```
function f(){
  var obj = document.getElementById( 'd1' );
  //DOM -> jQuery对象
  var $obj = $(obj);
  $obj.html( 'hello jQuery' );
}
```



1.3.3. 【jQuery 对象与 DOM 对象之间的转换】jQuery 对象 -> DOM 对象




jQuery对象 -> DOM对象

- 实现方式：

\$obj.get(0) 或者 \$obj.get()[0]

如：

```
function f(){
  var $obj = $( '#d1' );
  //jQuery对象 -> DOM
  var obj = $obj.get(0);
  obj.innerHTML = 'hello jQuery' ;
}
```



2. jQuery 选择器

2.1. jQuery 选择器

2.1.1. 【jQuery 选择器】什么是 jQuery 选择器

Technology
Tarena
达内科技

什么是jQuery选择器

- jQuery选择器类似于CSS选择（定位元素，施加样式），能够实现定位元素，施加行为
- 使用jQuery选择器能够将内容与行为分离

知识讲解

+

2.1.2. 【jQuery 选择器】选择器的种类

Technology
Tarena
达内科技

选择器的种类

- jQuery选择器包含如下种类：
 - 基本选择器
 - 层次选择器
 - 过滤选择器
 - 表单选择器

参考官方下载的jQuery文档

知识讲解

+

2.2. 基本选择器

2.2.1. 【基本选择器】#id

Technology
Tarena
达内科技

#id

- 特点：最快，尽量使用id选择器

```
<body>
  <div id= "d1" >hello jQuery</div>
</body>
```

```
function f(){
  $('#d1').css( 'font-size' , ' 30px' );
}
```

知识讲解

+

2.2.2. 【基本选择器】.class

Tarena
达内科技

.class

- 特点：根据class属性定位元素

```
<body>
  <div class= "s1" >hello</div>
  <div class= "s1" >jQuery</div>
</body>
```

```
function f(){
  $( '.s1' ).css( 'font-size' , ' 30px' );
}
```

++

知识归纳

2.2.3. 【基本选择器】element

Tarena
达内科技

element

- 标记选择器或元素选择器，依据HTML标记来区分

```
<body>
  <div id= "d1" >hello</div>
  <div class= "s1" >jQuery</div>
</body>
```

```
function f(){
  $( 'div' ).css( 'font-size' , ' 30px' );
}
```

++

知识归纳

2.2.4. 【基本选择器】selector1,selector2,...selectorN

Tarena
达内科技

selector1,selector2,...selectorN

- 合并选择器，即所有选择器的合集

```
<body>
  <div id= "d1" >hello</div>
  <div class= "s1" >jQuery</div>
  <p>空白格</p>
</body>
```

```
function f(){
  $( '#d1 , .s1 , p' )
    .css ( 'font-size' , ' 30px' );
}
```

++

知识归纳

2.2.5. 【基本选择器】*

Technology
Tarena
达内科技

* 所有页面元素，很少使用

```
<body>
  <div id= "d1" >hello</div>
  <div class= "s1" >jQuery</div>
  <p>空白格</p>
</body>
```

```
function f(){
    $( '*' ).css ( 'font-size' , ' 30px' );
}
```

知识讲解

2.3. 层次选择器

2.3.1. 【层次选择器】select1 空格 select2

Technology
Tarena
达内科技

select1 空格 select2

- 根据select1找到节点后，再去寻找子节点中符合select2的节点（重要）

```
<div id= "d1" >
  <div id= "d2" >hello</div>
  <div class= "s1" >jQuery</div>
  <p>空白格</p>
</div>
```

```
function f(){
    $( '#d1 div' )
      .css ( 'font-size' , ' 30px' );
}
```

知识讲解

2.3.2. 【层次选择器】select1 > select2

Technology
Tarena
达内科技

select1 > select2

- 只查找直接子节点，不查找间接子节点

```
<div id= "d1" >
  <div>hello</div>
  <p>jQuery</p>
  <div><p>空白格</p></div>
</div>
```

```
function f(){
    $( '#d1 > div' )
      .css ( 'font-size' , ' 30px' );
}
```

知识讲解

2.3.3. 【层次选择器】select1+select2

Technology
Tarena
达内科技

select1+select2

- “+” 表示下一个兄弟节点

```
<div id= "d1" >
  <div id= "d2" >hello</div>
  <div id= "d3" >jQuery</div>
</div><p id= "d4" >空白格</p></div>
```

```
function f(){
  $( '#d2 + div' )
    .css ( 'font-size' , ' 30px' );
}
```

+ +

2.3.4. 【层次选择器】select1~select2

Technology
Tarena
达内科技

select1~select2

- “~” 代表下面的所有兄弟

```
<div id= "d1" >
  <div id= "d2" >hello</div>
  <div id= "d3" >jQuery</div>
</div><p id= "d4" >空白格</p></div>
```

```
function f(){
  $( '#d2 ~ div' )
    .css ( 'font-size' , ' 30px' );
}
```

+ +

2.4. 过滤选择器

2.4.1. 【过滤选择器】基本过滤选择器

Technology
Tarena
达内科技


基本过滤选择器

- 过滤选择器以 “:” 或 “[]” 开始

- : first	第一个元素
- : last	最后一个元素
- : not (selector)	把selector排除在外
- : even	挑选偶数行
- : odd	挑选奇数行
- : eq (index)	下标等于index的元素
- : gt (index)	下标大于index的元素
- : lt (index)	下标小于index的元素

+ +

基本过滤选择器 (续1)




```
<table>
<thead>
<tr><td>歌手</td><td>年龄</td></tr>
</thead>
<tbody>
<tr><td>邓小棋</td><td>22</td></tr>
<tr id="tr1"><td>梦叔</td><td>31</td></tr>
<tr><td>王小二</td><td>26</td></tr>
<tr><td>雨神</td><td>26</td></tr>
</tbody>
</table>
```

```
function f(){
    $( 'table tr: first' )
        .css ( 'background-color' , ' #ccc' );
}
```

代码清单

++

基本过滤选择器 (续2)




```
function f(){
    $( 'table tr: first' )
        .css ( 'background-color' , ' #ccc' );
    $( 'tbody tr: eq(2)' )
        .css ( 'background-color' , ' #ccc' );
    $( 'tbody tr: even' )
        .css ( 'background-color' , ' #ccc' );
    $( 'tbody tr: odd' )
        .css ( 'background-color' , ' #red' );
    $( 'tbody tr: eq(1) td: eq(1)' )
        .css ( 'background-color' , ' #blue' );
    $( 'tbody tr: not(#tr1)' )
        .css ( 'background-color' , ' #ccc' );
}
```

代码清单

++

2.4.2. 【过滤选择器】内容过滤选择器

内容过滤选择器



• 内容过滤选择器包含：

- : contains (text) 匹配包含给定文本的元素
- : empty 匹配所有不包含子元素或文本的空元素
- : has (selector) 匹配含有选择器所匹配的元素
- : parent 匹配含有子元素或者文本的元素

代码清单

++

2.4.3. 【过滤选择器】可见性过滤选择器

Technology
Tarena
达内科技

可见性过滤选择器

- 可见性过滤选择器包含：
 - `: hidden` 匹配所有不可见元素，或type为hidden的元素
 - `: visible` 匹配所有的可见元素

+ +

Technology
Tarena
达内科技

可见性过滤选择器（续）

```
<div> hello jQuery </div>
<div style= "display:none;" > hello Ajax </div>
```

```
function f(){
    $( 'div : hidden' ) . css( 'display' , ' block' );
    $( 'div : hidden' ) . show(normal);
    $( 'div : visible' ) . hide(800);
}
```

+ +

2.4.4. 【过滤选择器】属性过滤选择器

Technology
Tarena
达内科技

属性过滤选择器

- 属性过滤器包含如下：
 - `{ attribute }` 匹配具有attribute属性的元素
 - `{ attribute = value }` 匹配属性等于value的元素
 - `{ attribute != value }` 匹配属性不等于value的元素

+ +

2.4.5. 【过滤选择器】子元素过滤选择器

Tarena
达内科技

子元素过滤选择器

- 子元素过滤选择器包括：
 - `:nth-child (index/even/odd)` 将为每一个父元素匹配子元素，index是从1开始的整数，表示对应位置的子元素
 - `:eq(index)`匹配一个给定索引值的元素，index是从0开始的整数

知识讲解

+

Tarena
达内科技

子元素过滤选择器（续）

```
<ul>
<li>item1</li>
<li>item2</li>
<li>item3</li>
</ul>
<ul>
<li>item11</li>
<li>item22</li>
<li>item33</li>
</ul>
```

\$('ul li: eq(1)')

\$('ul li: nth-child(2)')

知识讲解

+

2.4.6. 【过滤选择器】表单对象属性过滤选择器

Tarena
达内科技

表单对象属性过滤选择器

- 表单对象属性过滤选择器包括：
 - `: enabled`
 - `: disabled`
 - `: checked`
 - `: selected`

知识讲解

+

2.5. 表单选择器

2.5.1. 【表单选择器】表单选择器

知识讲解

表单选择器

• 表单选择器包括：

- :input	- :image
- :text	- :reset
- :password	- :button
- :radio	- :file
- :checkbox	- :hidden
- :submit	

3. jQuery 操作 DOM

3.1. jQuery 操作 DOM-查询

3.1.1. 【jQuery 操作 DOM-查询】html ()

知识讲解

html ()

• 读取或修改节点的HTML内容

```
<div id= 'd1' >
  <span> hello jQuery </span>
</div>
```

```
function f(){
  alert$( 'd1' ) . html( );
}
```

3.1.2. 【jQuery 操作 DOM-查询】text ()

知识讲解

text ()


• 读取或修改节点的文本内容

```
<div id= 'd1' >
  <span> hello jQuery </span>
</div>
```


```
function f(){
  alert$( 'd1' ) . text( );
}
```

80

3.1.3. 【jQuery 操作 DOM-查询】val ()


<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>val ()</h4> <ul style="list-style-type: none"> 读取或修改节点的value属性值 <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <code><input name= 'uname' /></code> </div> <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <pre>function f(){ alert(\$(' :text').val('空白格')); }</pre> </div> <div style="text-align: right;">++</div>
---	---

3.1.4. 【jQuery 操作 DOM-查询】attr ()


<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>attr ()</h4> <ul style="list-style-type: none"> 读取或者修改节点的属性 <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <code><div id= 'd1' > hello jQuery </div></code> </div> <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <pre>function f(){ \$(' #d1').attr('style' , ' color : red ; '); }</pre> </div> <div style="text-align: right;">++</div>
---	--

3.2. jQuery 操作 DOM-创建、插入、删除

3.2.1. 【jQuery 操作 DOM-创建、插入、删除】创建 DOM 节点


<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>创建DOM节点</h4> <ul style="list-style-type: none"> \$(html) 如创建一个div，并加在body的最后一个节点， 代码如下： <pre>var \$obj = \$('<div>hellojQuery</div>'); \$('body'). append (\$obj);</pre> <p>简写形式：</p> <pre>\$('body'). append ('<div>...</div>');</pre> <div style="text-align: right;">++</div>
---	---


3.2.2. 【jQuery 操作 DOM-创建、插入、删除】插入 DOM 节点




插入DOM节点

- append () 做为最后一个子节点添加进来
- prepend () 做为第一个子节点添加进来
- after () 做为下一个兄弟节点添加进来
- before () 做为上一个兄弟节点添加进来







3.2.3. 【jQuery 操作 DOM-创建、插入、删除】删除 DOM 节点



删除DOM节点


- remove () 移除
- remove (selector) 按选择器定位后删除
- empty () 清空节点






3.3. JQuery 操作 DOM – 复制节点


3.3.1. 【JQuery 操作 DOM – 复制节点】复制 DOM 节点



复制DOM节点



- clone ()
- clone (true) : 复制的节点也具有行为 (将处理代码一块复制)







3.4. JQuery 操作 DOM – 样式操作

3.4.1. 【JQuery 操作 DOM – 样式操作】样式操作

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4 style="text-align: center;">样式操作</h4> <ul style="list-style-type: none"> • attr ("class" , "") 获取和设置 • addClass (" ") 追加样式 • removeClass (" ") 移除样式 • removeClass () 移除所有样式 • toggleClass (" ") 切换样式 • hasClass (" ") 是否有某个样式 • css("") 读取css的值 • css (" " , " ") 设置多个样式 <div style="text-align: right;">  </div> <div style="text-align: right;">+</div>
---	---

3.5. JQuery 操作 DOM – 遍历节点

3.5.1. 【JQuery 操作 DOM – 遍历节点】遍历节点

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4 style="text-align: center;">遍历节点</h4> <ul style="list-style-type: none"> • children () / children (selector) 只考虑直接子节点 • next () / next (selector) 下一个兄弟节点 • prev () / prev (selector) 上一个兄弟节点 • siblings () / siblings (selector) 其他兄弟 • find (selector) 查找满足选择器的所有后代 • parent () 父节点 (没有选择器) <div style="text-align: right;">  </div> <div style="text-align: right;">+</div>
---	---

经典案例

1. jQuery 简单案例

- 问题

使用 jQuery 实现对 div 内容的样式修改。

- 方案

使用<script>标签引入 jQuery 的 js 文件，获取 div 的 jQuery 对象，调用 jQuery 方法修改样式。

- 步骤

步骤一：新建 selector/s01.html 文件

在页面内添加一个超链接，用于调用 JavaScript 的方法。添加一个 div 元素，填写一些文本内容。

```
<!-- jQuery简单案例 -->
<a id="a1" href="javascript:;"
    onclick="">hello jQuery</a>
<div id="d1">write less, do more</div>
</body>
```

随调用的方法不同而改变

图 - 1

步骤二：引入 jQuery.js 文件

```
<script type="text/javascript"
    src="../../js/jquery-1.11.1.js"></script>
```

图 - 2

步骤三：添加 f1 () 方法：修改 div 内容的传统方式

```
/*之前的做法：原始方式*/
function f1() {
    var obj = document.getElementById('a1');
    alert(obj.innerHTML);
}
```

图 - 3

步骤四：添加 f2 () 方法：使用 jQuery 的方法修改 div 内容

```
/*以后的做法:jQuery方式 */
function f2() {
    //将原始的DOM对象封装成一个jQuery对象
    var obj = $('#a1');
    //DOM对象的obj.innerHTML不能使用了。
    //要使用jQuery对象提供的方法或者属性。
    alert(obj.html());
}
```

图 - 4

步骤五：运行查看效果

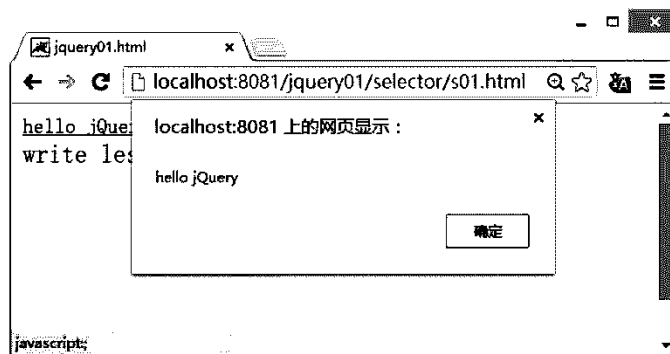


图 - 5

步骤六：添加 f4 () 方法：jQuery 方法的连缀调用

```
function f4() {
    $('#d1').css('font-size', '60px')
        .css('font-style', 'italic');
}
```

图 - 6

步骤七：运行查看结果

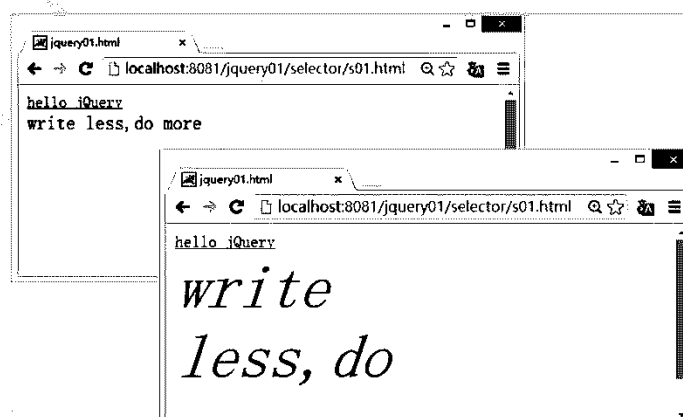


图 - 7

• 完整代码

s01.html 文件代码如下：

```
<html>
<head>
<title>jquery01.html</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<style>
    #d1{
        width:200px;
        height:200px;
        background-color:#fff8dc;
        font-size:20px;
    }
</style>
<script type="text/javascript"
    src="../js/jquery-1.11.1.js"></script>
<script>
    /*之前的做法:原始方式 */
    function f1(){
        var obj = document.getElementById('a1');
        alert(obj.innerHTML);
    }

    /*以后的做法:jQuery 方式 */
    function f2(){
        //将原始的 DOM 对象封装成一个 jQuery 对象
        var obj = $('#a1');
        //DOM 对象的 obj.innerHTML 不能使用了。
        //要使用 jQuery 对象提供的方法或者属性。
        alert(obj.html());
    }

    function f4(){
        $('#d1').css('font-size','60px')
        .css('font-style','italic');
    }
</script>
</head>
<body>
<!-- jQuery 简单案例 -->
<a id="a1" href="javascript:;"
    onclick="f4();">hello jQuery</a>
<div id="d1">write less,do more</div>
</body>
</html>
```

2. jQuery 对象与 DOM 对象的转换

- 问题

如何将 DOM 包装为 jQuery 对象，以及如何将 jQuery 对象转换为 DOM 对象。

- 方案

为 DOM 对象添加\$符号，为 jQuery 对象调用 get 方法。

• 步骤

步骤一：新建 s02.html 页面

在页面中添加两个链接用于调用方法。在页面中引入 jQuery 文件。

```
<script type="text/javascript"
    src="../../js/jquery-1.11.1.js"></script>
<body>
<!-- jQuery对象与DOM对象的转换 -->
<a id="a1" href="javascript:;" onclick="f1();" >
    DOM对象-->jQuery对象</a><br/><br/>
<a id="a1" href="javascript:;" onclick="f2();" >
    jQuery对象-->DOM对象</a><br/><br/>
```

图 - 8

步骤二：添加 f1() 方法：DOM 对象→jQuery 对象

```
//DOM对象转换成jQuery对象
function f1(){
    var obj = document.getElementById('a1');
    var $obj = $(obj);
    alert($obj.html());
}
```

图 - 9

步骤三：运行查看结果

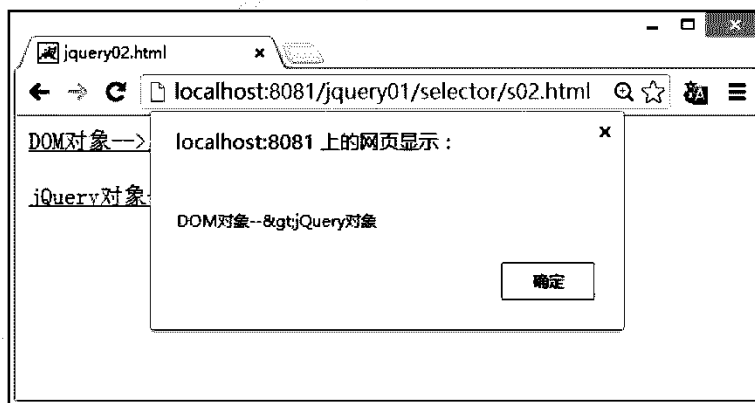


图 - 10

步骤四：添加 f2() 方法：jQuery 对象→DOM 对象

```
//jQuery对象转换成DOM对象
function f2(){
    var $obj = $('#a2');
    //var obj = $obj.get(0);
    var obj = $obj.get()[0];
    alert(obj.innerHTML);
}
```

图 - 11

步骤五：运行查看结果



图 - 12

• 完整代码

s02.html 文件代码如下：

```
<html>
<head>
<title>jquery02.html</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script type="text/javascript"
src="../js/jquery-1.11.1.js"></script>
<script>
//DOM 对象转换成 jQuery 对象
function f1(){
var obj = document.getElementById('a1');
var $obj = $(obj);
alert($obj.html());
}
//jQuery 对象转换成 DOM 对象
function f2(){
var $obj = $('#a2');
//var obj = $obj.get(0);
var obj = $obj.get()[0];
alert(obj.innerHTML);
}
</script>
</head>
<body>
<!-- jQuery 对象与 DOM 对象的转换 -->
<a id="a1" href="javascript:;" onclick="f1();">
DOM 对象-->jQuery 对象</a><br/><br/>
<a id="a2" href="javascript:;" onclick="f2();">
jQuery 对象-->DOM 对象</a><br/><br/>
</body>
</html>
```

3. jQuery 选择器

• 问题

掌握 jQuery 中的基本选择器、层次选择器、过滤选择器、表单选择器。

• 方案

引入 jQuery 文件，调用选择器方法，查看选择器的操作结果。

• 步骤

步骤一：基本选择器

新建 s03.html 页面：

```
<body>
<!-- jQuery基本选择器 -->
<!-- #id -->
<!-- .class -->
<!-- element -->
<!-- selector1,selector2,... -->
<!-- * -->
<div id="d1">Hello jQuery</div>
<div class="s1">Hello Ajax</div>
<p class="s1">Hello Servlet</p>
<a href="javascript:;" onclick=" " >基本选择器</a>
</body>
```

图 - 13

引入 jQuery 文件(后面案例中，该步骤会省略，但所有案例都必须引入 jQuery 文件方可运行)。

```
<script type="text/javascript"
src="../js/jquery-1.11.1.js"></script>
```

图 - 14

添加 f1(), f2(), f3(), f4(), f5() 方法，进行各基本选择器的调用，查看结果。

```
<script type="text/javascript">
function f1(){
    $('#d1').css('font-size','30px');
}
function f2(){
    $('.s1').css('font-size','30px');
}
function f3(){
    $('div').css('font-size','30px');
}
function f4(){
    $('#d1,p').css('font-size','30px');
}
function f5(){
    $('*').css('font-size','30px');
}
```

图 - 15

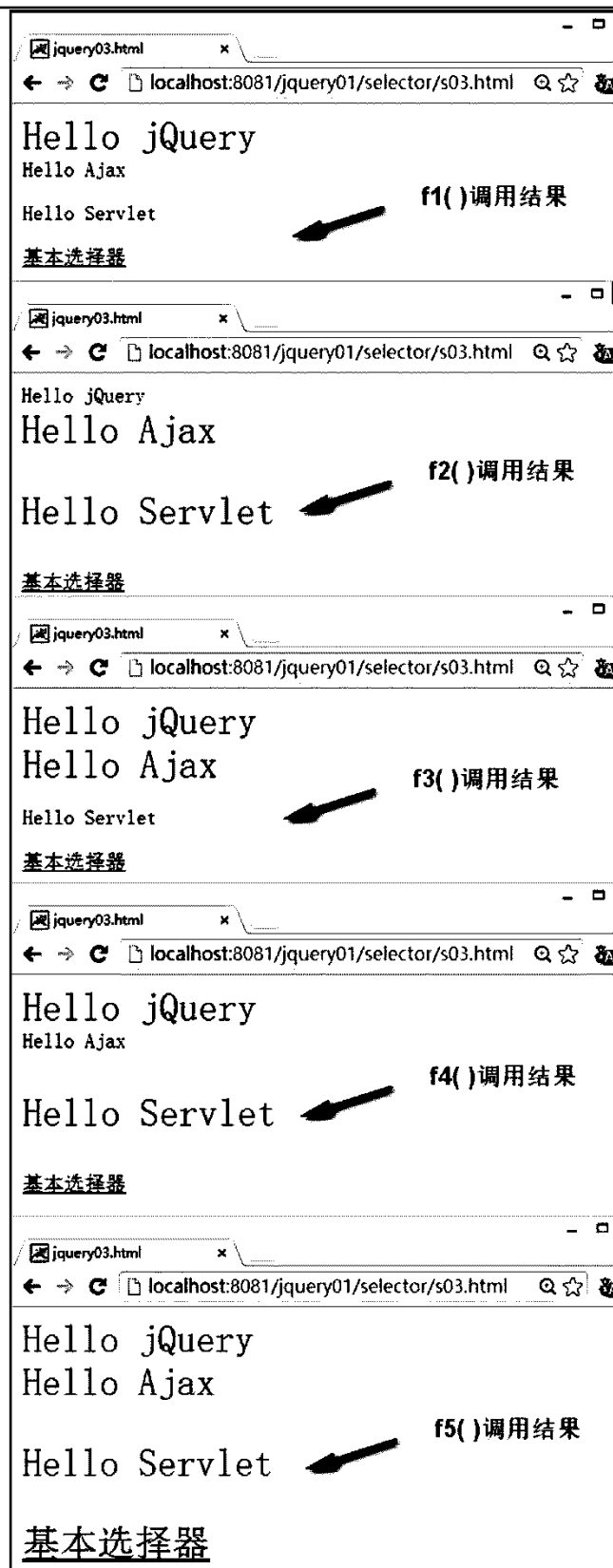


图 - 16

步骤二：层次选择器

新建 s04.html 页面：

```
<body>
<!-- jQuery层次选择器 -->
<!-- select1 空格 select2 -->
<!-- select1 > select2 -->
<!-- select1 + select2 -->
<!-- select1 ~ select2 -->
<div id="d1">
    <div id="d2">Hello Java</div>
    <div id="d3">
        <div id="d4" style="font-size:40px">Hello Servlet</div>
    </div>
    <div id="d5">Hello Ajax</div>
    <p>Hello jQuery</p>
</div>
<a href="javascript:;" onclick="f1();">基本选择器</a>
</body>
```

图- 17

添加方法：

```
function f1(){
    //根据select1找到点后,再去寻找子节点中符合select2的节点
    $('#d1 div').css('font-size','30px');
}
function f2(){
    //只查找子节点,不找非直接节点
    $('#d1>div').css('font-size','30px');
}

function f3(){
    //+表示下一个兄弟,
    $('#d3+div').css('font-size','30px');
    //不会有改变, p标记不是d3的兄弟节点
    $('#d3+p').css('font-size','30px');
}
function f4(){
    //~ 下面所有兄弟
    $('#d2~div').css('font-size','30px');
}
}
```

图 - 18

查看运行结果：



图 -19

步骤三：过滤选择器

基本过滤选择器：

新建 s05.html 页面：

```
<body style="font-size:24px">
  <!-- 基本过滤选择器 -->
  <!-- : first      第一行被挑选出来
       : last      最后一个
       : not(selector) 把selector排除在外
       : even      偶数行
       : odd       奇数行
       : eq(index) 如果要挑选第二行?
                   使用该方法, index是从0开始
       : gt(index) 大于index下标的
       : lt(index) 小于index下标的
  -->
  <a href="javascript:;" onclick="f4();" >过滤选择器</a>
  <table width="60%" border="1" cellpadding="0"
        cellspacing="0">
    <thead>
      <tr><td>name</td><td>age</td></tr>
    </thead>
    <tbody>
      <tr><td>Tom</td><td>21</td></tr>
      <tr id="tr2"><td>Jerry</td><td>22</td></tr>
      <tr><td>Kitty</td><td>23</td></tr>
      <tr><td>Doro</td><td>24</td></tr>
    </tbody>
  </table>
</body>
```

图 - 20

添加方法：

```
function f1(){
  $('table tr').css('background-color','#cccccc');
  $('table tr:first').css('background-color','red');
  $('tbody tr:eq(3)').css('background-color','yellow');
}
function f2(){
  $('tbody tr:even').css('background-color','#fff8dc');
  $('tbody tr:odd').css('background-color','#yellow');
}
function f3(){
  $('tbody tr:eq(1) td:eq(1)')
  .css('background-color','yellow')
  .css('font-size','30px');
}
function f4(){
  $('tbody tr:not(#tr2)')
  .css('background-color','yellow');
}
```

图 - 21

运行查看结果：



图 - 22

内容过滤选择器：

新建 s06.html 文件：

```
<body>
  <!-- 内容过滤选择器 -->
  <!-- : contains (text) 匹配包含给定文本的元素
       : empty 匹配所有不包含子元素或者文本的空
       : has (selector) 匹配含有选择器所匹配的元素
       : parent 匹配含有子元素或者文本的元素
  -->
  <a href="javascript:;" onclick="f4();" >
    内容过滤选择器</a><br><br>
  <div>这里是div</div>
  <div></div>
  <div>
    <p>含有p标记的div</p>
  </div>
</body>
```

图 - 23

添加方法：

```
<script type="text/javascript">
  function f1(){
    $('div:contains(这里)').css('font-size','30px');
  }
  function f2(){
    $('div:empty').html('空的div');
  }
  function f3(){
    $('div:has(p)').css('font-size','30px');
  }
  function f4(){
    $('div:parent').css('font-size','30px');
  }
</script>
```

图 - 24

运行结果：



图 - 25



图 - 26

可见性过滤选择器：

新建 s07.html 文件：

```
<body>
<!-- 可见性过滤选择器 -->
<!-- : hidden 匹配所有不可见元素，或者type为hidden的元素
      : visible 匹配所有的可见元素
-->
<a href="javascript:;" onclick="f2();" >可见性过滤选择器</a>
<div>这里是div</div>
<div style="display:none;">display:none</div>
</body>
```

图 - 27

添加方法：

```
<script type="text/javascript">
function f1(){
    $('div:hidden').css('display','block');
    $('div:hidden').show('normal');
}
function f2(){
    $('div:visible').hide(800);
}
</script>
```

图 - 28

运行结果：



图 - 29

属性过滤选择器：

新建 s08.html 文件：

```
<body>
  <!-- 属性过滤选择器 -->
  <!--      [attribute ]
            [attribute=value]
            [attribute !=value]
  -->
  <a href="javascript:;" onclick="f3();" >属性过滤选择器</a>
  <div id="d1">有Id的div: d1</div>
  <div id="d2">有Id的div: d2</div>
  <div>没有Id的div</div>
</body>
```

图 - 30

添加方法：

```
function f1(){
    $('div[id]').css('font-size','30px');
}
function f2(){
    $('div[id=d2]').css('font-size','30px');
}
function f3(){
    $('div[id!=d2]').css('font-size','30px');
}
```

图 - 31

运行结果：



图 - 32

子元素过滤选择器：

新建 s09.html 文件：

```
<!-- 子元素过滤选择器 -->
<!-- : nth-child(index / even / odd)
      index是从1开始的整数，表示对应位置的子元素
-->
<a href="javascript:;" onclick="f1();" >子元素过滤选择器</a>
<ul>
  <li>item1</li>
  <li>item2</li>
  <li>item3</li>
</ul>
<ul>
  <li>item111</li>
  <li>item222</li>
  <li>item333</li>
</ul>
```

图 - 33

添加方法：

```
<script type="text/javascript">
    function f1(){
        $('ul li:eq(1)')
            .css('font-size','30px');
    }
    function f2(){
        $('ul li:nth-child(2)')
            .css('font-size','30px');
    }
</script>
```

图 - 34

运行结果：

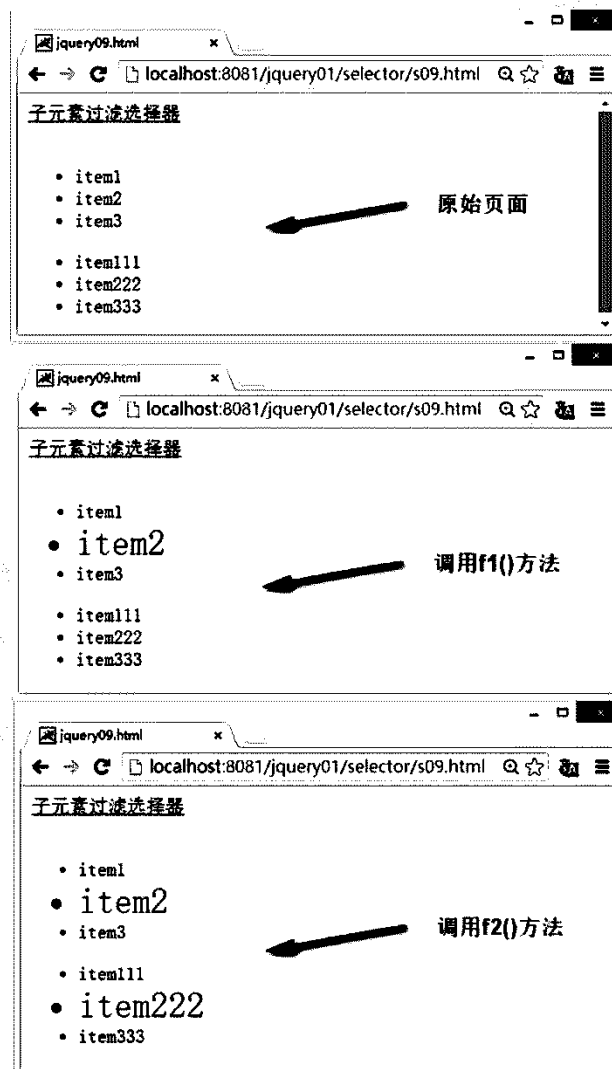


图 - 35

表单对象属性过滤选择器

新建 s10.html 文件：

```
<body>
<!-- 表单对象属性过滤器 -->
<!-- : enabled
      : disable
      : checked
      : selected
-->
<a href="javascript:;" onclick="f4();" >
    表单对象属性过滤器</a><br><br>
<form id="form1">
    username:<input name="username"/><br/>
    name:<input name="name" disabled="disabled"/><br/>
</form>
<form id="form2">
    兴趣: 吃饭<input type="checkbox" name="interest"
    value="eating" checked="checked">
    喝茶<input type="checkbox" name="interest"
    value="drinking" >
    看报纸<input type="checkbox" name="interest"
    value="reading" >
</form>
<form id="form3">
    <select>
        <option value="math">数学</option>
        <option value="english"
            selected="selected">英语</option>
        <option value="computer">计算机</option>
    </select>
</form>
</body>
```

图 - 36

添加方法：

```
<script type="text/javascript">
    function f1(){
        $('#form1 input:disabled')
            .css('border','1px dotted red');
    }
    function f2(){
        //$('#form1 input:enabled')
        //.css('border','1px dotted red');
        $('#form1 input:enabled')
            .attr('disabled',true);
    }
    function f3(){
        var length =
            $('#form2 input:checked').length;
        alert(length);
        alert($('#form2 input:checked').val());
    }
    function f4(){
        alert($('#form3 select option:selected').val());
        alert($('#option:selected').val());
    }
</script>
```

图 - 37

运行结果：



图 - 38

步骤四：选择器的综合案例

新建 e01.html 页面：

```
<body onload="f1();">
  <table width="50%" border="1" cellpadding="0"
    cellspacing="0">
    <caption style="font-weight:800;">员工信息</caption>
    <thead>
      <tr><th>姓名</th><th>薪水</th><th>年龄</th></tr>
    </thead>
    <tbody>
      <tr><td>张三</td><td>20000</td><td>23</td></tr>
      <tr><td>李四</td><td>22000</td><td>22</td></tr>
      <tr><td>王五</td><td>14000</td><td>26</td></tr>
      <tr><td>马六</td><td>15000</td><td>21</td></tr>
    </tbody>
  </table>
  <input type="button" value="Click1" onclick="f2();" />
  <input type="button" value="Click2" onclick="f3();" />
</body>
```

图 - 39

添加方法：

```
function f1(){
  $('tbody tr:even').css('background-color',
    '#fff8dc');
  $('tbody tr:odd').css('background-color',
    'yellow');
}

function f2(){
  $('tbody tr:contains(王五)').css(
    'background-color', 'yellow');
}

function f3(){
  $('tbody tr:eq(1) td:eq(1)').css(
    'background-color', 'red');
}
```

图 - 40

查看运行结果：

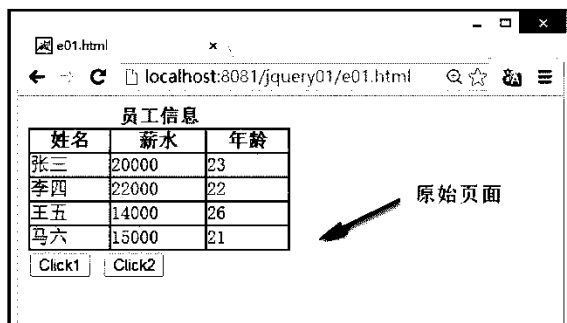


图 - 41

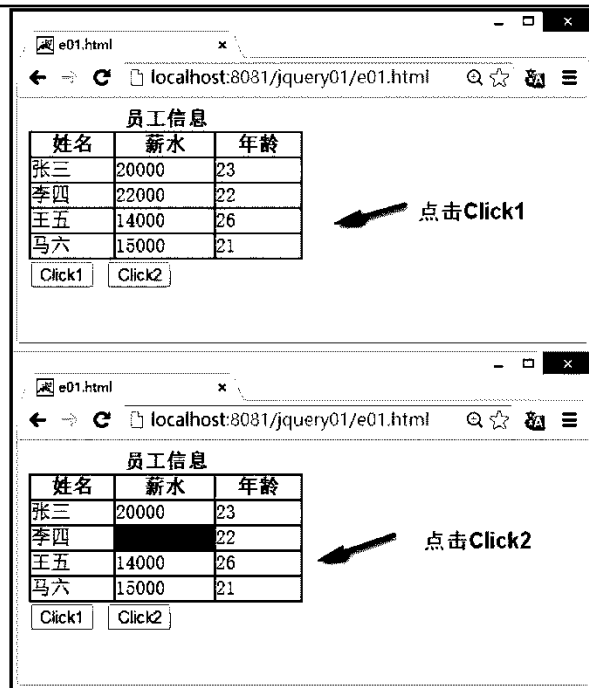


图 - 42

• 完整代码

s03.html 文件代码：

```
<html>
<head>
<title>jquery03.html</title>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <script type="text/javascript"
    src="../../js/jquery-1.11.1.js"></script>
  <script type="text/javascript">
    function f1(){
      $('#d1').css('font-size','30px');
    }
    function f2(){
      $('.s1').css('font-size','30px');
    }
    function f3(){
      $('div').css('font-size','30px');
    }
    function f4(){
      $('#d1,p').css('font-size','30px');
    }
    function f5(){
      $('*').css('font-size','30px');
    }
  </script>
</head>
<body>
<!-- jQuery 基本选择器 -->
<!-- #id -->
<!-- .class -->
<!-- element -->
<!-- selector1,selector2,... -->
<!-- * -->
<div id="d1">Hello jQuery</div>
```

```
<div class="s1">Hello Ajax</div>
<p class="s1">Hello Servlet</p>
<a href="javascript:;" onclick="f5()">基本选择器</a>
</body>
</html>
```

s04.html 文件代码：

```
<html>
<head>
<title>jquery04.html</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script type="text/javascript" src="../../js/jquery-1.11.1.js"></script>
<script type="text/javascript">
    function f1(){
        $('#d1 div').css('font-size','30px');
    }
    function f2(){
        //只查找子节点，不找非直接节点
        //只找到 d2, d3, d5.但是 d4 继承了 d3, 也会改变
        $('#d1>div').css('font-size','30px');
    }

    function f3(){
        //+表示下一个兄弟,
        $('#d3+div').css('font-size','30px');
        //不会有改变, p 标记不是 d3 的兄弟节点
        $('#d3+p').css('font-size','30px');
    }
    function f4(){
        //~ 下面所有兄弟
        $('#d2~div').css('font-size','30px');
    }
</script>
</head>
<body>
<!-- jQuery 层次选择器 -->
<!-- select1 空格 select2 -->
<!-- select1 > select2 -->
<!-- select1 + select2 -->
<!-- select1 ~ select2 -->
<div id="d1">
    <div id="d2">Hello Java</div>
    <div id="d3">
        <div id="d4" style="font-size:40px">Hello Servlet</div>
    </div>
    <div id="d5">Hello Ajax</div>
    <p>Hello jQuery</p>
</div>
<a href="javascript:;" onclick="f4();">基本选择器</a>
</body>
</html>
```

s05.html 文件代码：

```
<html>
<head>
<title>jquery05.html</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
```

```
<script type="text/javascript" src="../js/jquery-1.11.1.js"></script>
<script type="text/javascript">
    function f1(){
        $('table tr').css('background-color','#cccccc');
        $('table tr:first').css('background-color','red');
        $('tbody tr:eq(3)').css('background-color','yellow');
    }
    function f2(){
        $('tbody tr:even').css('background-color','#fff8dc');
        $('tbody tr:odd').css('background-color','#yellow');
    }

    function f3(){
        $('tbody tr:eq(1) td:eq(1)')
        .css('background-color','yellow')
        .css('font-size','30px');
    }
    function f4(){
        $('tbody tr:not(#tr2)')
        .css('background-color','yellow');
    }
</script>
</head>

<body style="font-size:24px">
<!-- 基本过滤选择器 -->
<!-- :first      第一行被挑选出来
      :last      最后一个
      :not(selector) 把 selector 排除在外
      :even      偶数行
      :odd       奇数行
      :eq ( index ) 如果要挑选第二行？
                  使用该方法，index 是从 0 开始
      :gt ( index ) 大于 index 下标的
      :lt ( index ) 小于 index 下标的
-->
<a href="javascript:;" onclick="f4();" >过滤选择器</a>
<table width="60%" border="1" cellpadding="0"
      cellspacing="0">
  <thead>
    <tr><td>name</td><td>age</td></tr>
  </thead>
  <tbody>
    <tr><td>Tom</td><td>21</td></tr>
    <tr id="tr2"><td>Jerry</td><td>22</td></tr>
    <tr><td>Kitty</td><td>23</td></tr>
    <tr><td>Doro</td><td>24</td></tr>
  </tbody>
</table>
</body>
</html>
```

s06.html 文件代码：

```
<html>
<head>
  <title>jquery06.html</title>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <script type="text/javascript" src="../js/jquery-1.11.1.js"></script>
```



```
<script type="text/javascript">
    function f1(){
        $('div:contains(这里)').css('font-size','30px');
    }
    function f2(){
        $('div:empty').html('空的 div');
    }
    function f3(){
        $('div:has(p)').css('font-size','30px');
    }
    function f4(){
        $('div:parent').css('font-size','30px');
    }
</script>
</head>

<body>
    <!-- 内容过滤选择器 -->
    <!-- :contains ( text ) 匹配包含给定文本的元素
        :empty 匹配所有不包含子元素或者文本的空
        :has ( selector ) 匹配含有选择器所匹配的元素
        :parent 匹配含有子元素或者文本的元素
    -->
    <a href="javascript:;" onclick="f4();" >
        内容过滤选择器</a><br><br>
    <div>这里是 div</div>
    <div></div>
    <div>
        <p>含有 p 标记的 div</p>
    </div>
</body>
</html>
```

s07.html 文件代码：

```
<html>
<head>
    <title>jquery07.html</title>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <script type="text/javascript" src="../js/jquery-1.11.1.js"></script>
    <script type="text/javascript">
        function f1(){
            // $('div:hidden').css('display','block');
            $('div:hidden').show('normal');
        }
        function f2(){
            $('div:visible').hide(800);
        }
    </script>
</head>

<body>
    <!-- 可见性过滤选择器 -->
    <!-- :hidden 匹配所有不可见元素，或者 type 为 hidden 的元素
        :visible 匹配所有的可见元素
    -->
    <a href="javascript:;" onclick="f2();" >可见性过滤选择器</a><br><br>
    <div>这里是 div</div>
```

```
<div style="display:none;">display:none</div>
</body>
</html>
```

s08.html 文件代码：

```
<html>
<head>
<title>jquery08.html</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script type="text/javascript" src="../js/jquery-1.11.1.js"></script>
<script type="text/javascript">
    function f1(){
        $('div[id]').css('font-size','30px');
    }
    function f2(){
        $('div[id=d2]').css('font-size','30px');
    }
    function f3(){
        $('div[id!=d2]').css('font-size','30px');
    }
</script>
</head>

<body>
<!-- 属性过滤选择器 -->
<!-- [attribute ]
      [attribute=value]
      [attribute !=value]
-->
<a href="javascript:;" onclick="f3();">属性过滤选择器</a><br><br>
<div id="d1">有 Id 的 div:d1</div>
<div id="d2">有 Id 的 div:d2</div>
    <div>没有 Id 的 div</div>
</body>
</html>
```

s09.html 文件代码：

```
<html>
<head>
<title>jquery09.html</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script type="text/javascript" src="../js/jquery-1.11.1.js"></script>
<script type="text/javascript">
    function f1(){
        $('ul li:eq(1)')
            .css('font-size','30px');
    }
    function f2(){
        $('ul li:nth-child(2)')
            .css('font-size','30px');
    }
</script>
</head>

<body>
<!-- 子元素过滤选择器 -->
<!-- :nth-child(index / even / odd)
      index 是从 1 开始的整数，表示对应位置的子元素
-->
```

```
-->
<a href="javascript:;" onclick="f2();" >子元素过滤选择器</a><br><br>
<ul>
  <li>item1</li>
  <li>item2</li>
  <li>item3</li>
</ul>
<ul>
  <li>item111</li>
  <li>item222</li>
  <li>item333</li>
</ul>
</body>
</html>
```

s10.html 文件代码：

```
<html>
<head>
  <title>jquery10.html</title>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <script type="text/javascript" src="../js/jquery-1.11.1.js"></script>
  <script type="text/javascript">
    function f1(){
      $('#form1 input:disabled')
        .css('border','1px dotted red');
    }
    function f2(){
      //$('#form1 input:enabled')
      //$.css('border','1px dotted red');
      $('#form1 input:enabled')
        .attr('disabled',true);
    }
    function f3(){
      var length =
        $('#form2 input:checked').length;
      alert(length);
      alert($('#form2 input:checked').val());
    }
    function f4(){
      alert($('#form3 select option:selected').val());
      alert($('#option:selected').val());
    }
  </script>
</head>

<body>
  <!-- 表单对象属性过滤选择器 -->

  <!-- :enabled
      :disable
      :checked
      :selected
-->
  <a href="javascript:;" onclick="f4();" >
    表单对象属性过滤选择器</a><br><br>
  <form id="form1">
    username:<input name="username"/><br>
    name:<input name="name" disabled="disabled"/><br>
  </form>
  <form id="form2">
    兴趣:吃饭<input type="checkbox" name="interest"
      value="eating" checked="checked">
```

```

喝茶<input type="checkbox" name="interest"
value="drinking" >
看报纸<input type="checkbox" name="interest"
value="reading" >
</form>
<form id="form3">
  <select>
    <option value="math">数学</option>
    <option value="english"
      selected="selected">英语</option>
    <option value="computer">计算机</option>
  </select>
</form>
</body>
</html>

```

e01.html 文件代码：

```

<html>
<head>
  <title>e01.html</title>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
  <script type="text/javascript" src="js/jquery-1.11.1.js"></script>
  <script type="text/javascript">
    function f1(){
      $('tbody tr:even').css('background-color',
        '#fff8dc');
      $('tbody tr:odd').css('background-color',
        'yellow');
    }

    function f2(){
      $('tbody tr:contains(王五)').css(
        'background-color','yellow');
    }

    function f3(){
      $('tbody tr:eq(1) td:eq(1)').css(
        'background-color','red');
    }
  </script>
</head>
<body onload="f1();">
  <table width="50%" border="1" cellpadding="0"
    cellspacing="0">
    <caption style="font-weight:800;">员工信息</caption>
    <thead>
      <tr><th>姓名</th><th>薪水</th><th>年龄</th></tr>
    </thead>
    <tbody>
      <tr><td>张三</td><td>20000</td><td>23</td></tr>
      <tr><td>李四</td><td>22000</td><td>22</td></tr>
      <tr><td>王五</td><td>14000</td><td>26</td></tr>
      <tr><td>马六</td><td>15000</td><td>21</td></tr>
    </tbody>
  </table>
  <input type="button" value="Click1" onclick="f2();"/>
  <input type="button" value="Click2" onclick="f3();"/>
</body>
</html>

```

4. jQuery 操作 DOM

- 问题

使用 jQuery 对 DOM 节点进行增删改查追加等操作。

- 方案

使用 jQuery 中关于 DOM 操作的有关方法。实现 DOM 的查询、创建、插入节点、删除节点、复制节点、属性操作、样式操作、遍历节点。

- 步骤

步骤一：DOM 操作：查询

新建 d01.html 文件：

```
<body>
  <!-- DOM操作:查询 -->
  <!--
    找到节点之后，可以读取或者修改节点的
    HTML内容、文本内容、value属性值和属性值
  -->
  <div id="d1"><span>Hello jQuery</span></div>
  username:<input name="username"/><br/>
  <a href="javascript:;" onclick="f1();">DOM操作</a>
</body>
```

图 - 43

添加方法：

```
/*  html()    innerHTML
   text()    innerText
*/
function f1(){
    //alert($('#d1').html());
    alert($('#d1').text());
}

function f2(){
    $('#d1').html('<p>hello dojo</p>');
}
```

图 - 44

```
/*val() value*/
function f3(){
    //alert($('text').val());
    $('text').val('kitty');
}

function f4(){
    //alert($('#d1').attr('id'));
    $('#d1').attr('style','color:red
    ;font-style:italic;');
}
```

图 - 45

运行查看结果：

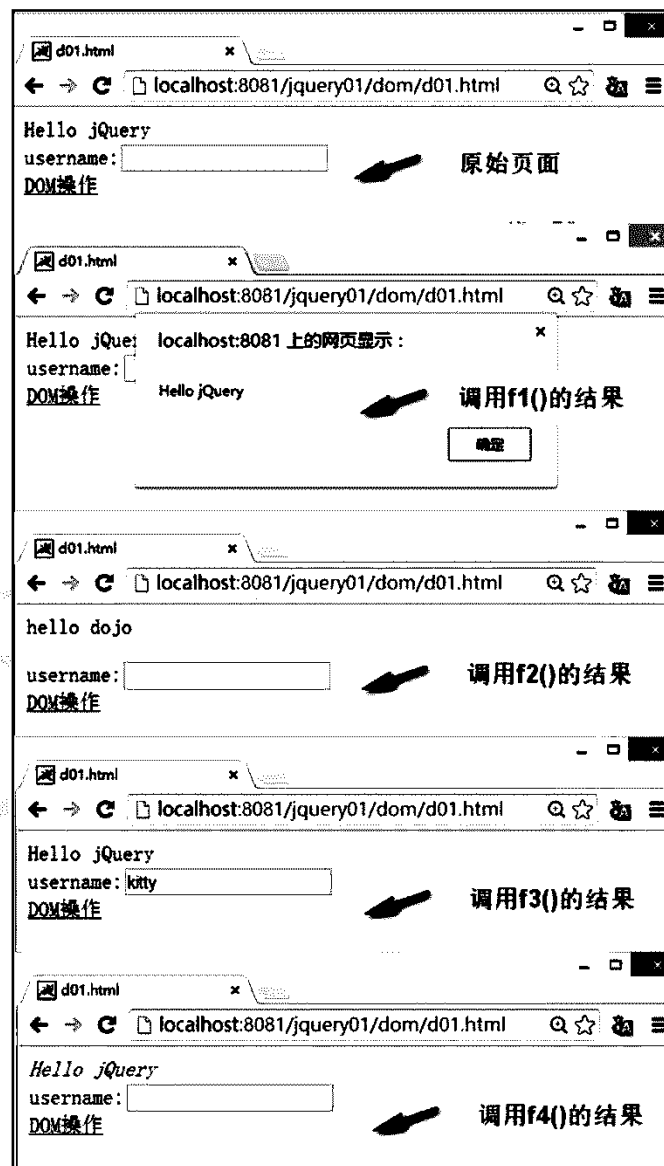


图 - 46

步骤二：DOM 操作：创建、插入、删除

新建 d02.html 文件：

```
<!--
    创建：$(html)
    插入节点：
        append()：作为最后一个孩子添加进来。
        prepend()：作为第一个孩子添加进来。
        after()：作为下一个兄弟添加进来
        before()：作为上一个兄弟添加进来
    删除节点：
        remove()
        remove(selector)
        empty()：清空节点
-->
<a href="javascript:;"
    onclick="f2();" >DIV的内容在哪里?</a>
<ul>
    <li>item1</li>
    <li id="l2">item2</li>
    <li>item3</li>
</ul>
<input type="button" value="插入"
    onclick="f3();" />
<input type="button" value="删除"
    onclick="f4();" />
</body>
```

图 - 47

添加方法：

```
function f1(){
    var $obj =
    $('<div>猜对了，在这里</div>');
    $('body').append($obj);
    //$('body').prepend($obj);
}
function f2(){
    $('body').append(
    '<div>猜对了，在这里</div>');
}
```

图 - 48

```
function f3(){
    //$('#ul').after('<p>hello jQuery</p>');
    $('#ul').before('<p>hello jQuery</p>');
}
function f4(){
    //$('#ul li:eq(1)').remove();
    $('#ul li:eq(1)').empty();
}
function f5(){
    $('#ul li').remove('#12');
}
```

图 - 49

运行查看结果：

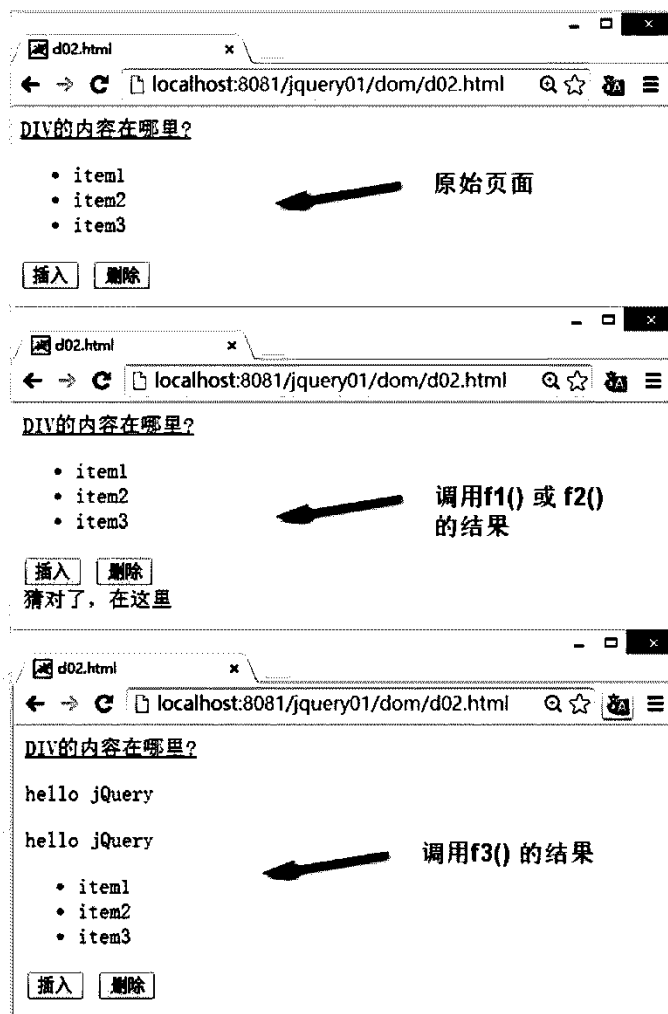


图 - 50



图 - 51

步骤三：分离 JavaScript 代码

分别使用传统方法和 jQuery 方法，在页面加载时就执行一段 JavaScript 代码。并将这些代码分离到一个 js 文件中。

新建 d03.html 文件：

```
<html>
<head>
<meta http-equiv="Content-Type"
      content="text/html; charset=UTF-8">
<title>Insert title here</title>
<script type="text/javascript" src="../js/j03.js">
</script>
</head>
<body>
    <div id="d1">hello jQuery</div>
</body>
</html>
```

图 - 52

新建 j03.js 文件：

```
window.onload = function(){
    var obj = document.getElementById('d1');
    obj.onclick=function(){
        this.innerHTML = 'hello Ajax';
    };
};
```

图 - 53

新建 d04.html 页面：

```
<html>
  <head>
    <meta http-equiv="Content-Type"
      content="text/html; charset=UTF-8">
    <title>Insert title here</title>
    <script type="text/javascript"
      src="../js/jquery-1.11.1.js"></script>
    <script type="text/javascript"
      src="../js/j04.js"></script>
  </head>
<body>
  <div id="d1">hello jQuery</div>
</body>
</html>
```

图 - 54

新建 j04.js 文件

```
$(function() {
  $('#d1').click(function() {
    $(this).html('hello Ajax');
  });
});
```

图 - 55

d03.html 文件和 d04.html 文件完成的操作是等价的。运行结果相同，都会在页面加载后，为 div 添加了点击事件，点击之后，div 内的文字会变更为 “hello Ajax”。

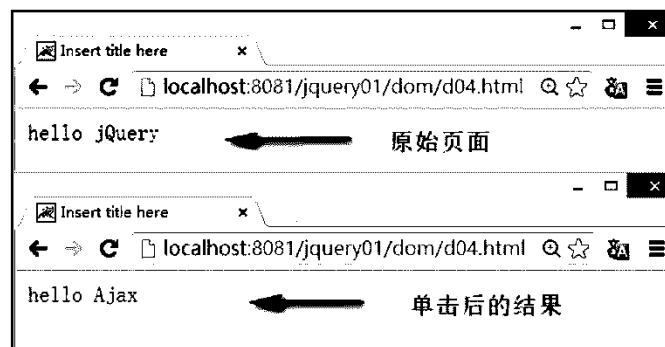


图 - 56

步骤四：复制节点

新建 d05.html 文件：

```
<!-- 复制节点 -->
<!--
    clone() :
    clone(true) : 使复制的节点也具有行为
                  (将处理代码一块复制)
-->
<ul>
    <li>item1</li>
    <li>item2</li>
    <li>item3</li>
</ul>
<a href="javascript:;" id="a1">复制节点</a>
```

图 - 57

添加方法：

```
$(function(){
    $('ul li:eq(2)').click(function(){
        $(this).css('font-size','30px');
    });

    $('#a1').click(function(){
        var $obj = $('ul li:eq(2)').clone();
        $('ul').append($obj);
    });
});
```

图 - 58

运行查看结果：

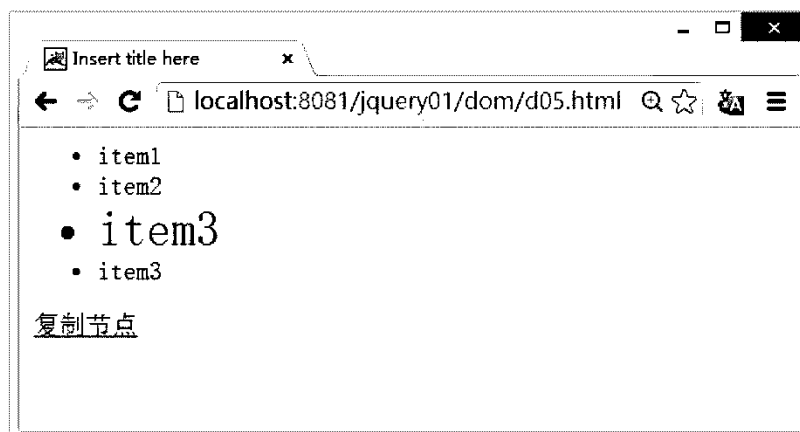


图 - 59

复制完的 item3 节点不具备原节点同样的单击事件。

```
$(function() {
    $('ul li:eq(2)').click(function() {
        $(this).css('font-size', '30px');
    });

    $('#a1').click(function() {
        var $obj = $('ul li:eq(2)').clone(true);
        $('ul').append($obj);
    });
});
```

图 - 60

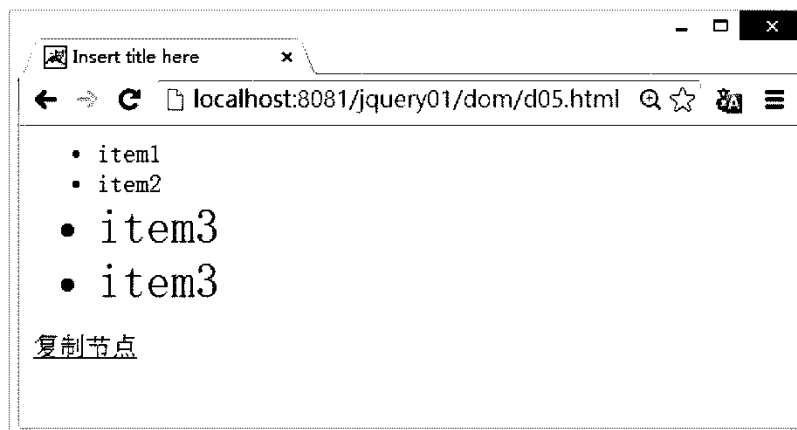


图 - 61

clone 方法添加了 true 参数以后，复制的节点也具有同原始节点相同的点击事件。

步骤五：样式操作

新建 d06.html 文件：

```
<body>
  <!-- 样式操作 -->
  <!--
    获取和设置: attr("class" , " ") 或者 attr("style", " ");
    追加: addClass(' ')
    移除: removeClass(' ') 或者
          removeClass('s1 s2') 或者
          removeClass( ) 会删除所有样式
    切换样式: toggleClass, 有该样式就删除, 没有就添加
    是否有某个样式 hasClass(' ')
    读取css(' ')
    设置css ( ' ', ' ' )
    或者css ({ ' ': ' ', ' ': ' ' }) 设置多个样式
  -->
  <div id="d1">hello jQuery</div>
  <a href="javascript:;" id="a1">样式操作</a>
</body>
```

图 - 62

添加方法：

```
<style>
  .s1{
    color:red;
  }
  .s2{
    font-size:20px;
  }
</style>
<script type="text/javascript"
  src="../../js/jquery-1.11.1.js"></script>
<script type="text/javascript">
  $(function(){
    $('#a1').click(function(){
      $('#d1').attr('style','color:red;font-size:30px;');
      $('#d1').attr('class','s1');
      $('#d1').addClass('s1');
      $('#d1').addClass('s1 s2');
      $('#d1').removeClass('s1');
      $('#d1').toggleClass('s2');
      alert($('#d1').hasClass('s1'));
      $('#d1').css('font-style','italic');
    });
  });
</script>
```

图 - 63

运行查看结果：

参考浏览器的实际运行效果。

步骤六：遍历节点

新建 d07.html 文件：

```
<!-- 遍历节点 -->
<!--
  children () / children(selector):
    只考虑子元素，不考虑其它后代
  next () / next (selector) : 下一个兄弟
  prev () / prev(selector): 上一个兄弟
  siblings () /siblings(selector): 其它兄弟
  find (selector) : 查找满足选择器的所有后代
  parent () : 父节点（没有选择器）
-->
<div id="d1">
  <div id="d2">hello 1</div>
  <div id="d3">hello 2</div>
  <div id="d4">hello 3</div>
  <p>hello 4</p>
</div>
<a href="javascript:;" id="a1">遍历节点</a>
```

图 - 64

添加方法：

```
$(function(){
    $('#a1').click(function(){
        var $obj = $('#d1').children('div');
        alert($obj.length);
        $obj.css('font-size','60px');

        $('#d3').next().css('font-size','60px');
        $('#d3').next('p').css('font-size','60px');

        $('#d3').prev().css('font-size','60px');

        $('#d3').siblings().css('font-size','60px');
        $('#d3').siblings('div').css('font-size','60px');

        $('#d1').find('p').css('font-size','60px');

        alert($('#d3').parent().attr('id'));
    });
});
```

图 - 65

运行查看结果：

逐个方法运行查看结果，以浏览器实际运行结果为准。

• 完整代码

d01.html 文件代码如下：

```
<html>
  <head>
    <title>d01.html</title>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <script type="text/javascript" src="../js/jquery-1.11.1.js"></script>
    <script type="text/javascript">
      /* html() innerHTML
       text() innerText
      */
      function f1(){
        //alert($('#d1').html());
        alert($('#d1').text());
      }

      function f2(){
        $('#d1').html('<p>hello dojo</p>');
      }

      /*val() value*/
      function f3(){
        //alert($('#:text').val());
        $('#:text').val('kitty');
      }

      function f4(){
        //alert($('#d1').attr('id'));
        $('#d1').attr('style','color:red;font-style:italic;');
      }
    </script>
  </head>
  <body>
    <!-- DOM操作:查询 -->
    <!--
```

找到节点之后，可以读取或者修改节点的

HTML 内容、文本内容、value 属性值和属性值

```
-->
<div id="d1"><span>Hello jQuery</span></div>
username:<input name="username"/><br/>
<a href="javascript:;" onclick="f4();" >DOM 操作</a>
</body>
</html>
```

d02.html 文件代码如下：

```
<html>
<head>
<title>d02.html</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script type="text/javascript" src="../js/jquery-1.11.1.js"></script>
<script type="text/javascript">
function f1(){
    var $obj =
        $('<div>猜对了，在这里</div>');
    $('body').append($obj);
    // $('body').prepend($obj);
}
function f2(){
    $('body').append(
        '<div>猜对了，在这里</div>');
}
function f3(){
    // $('ul').after('<p>hello jQuery</p>');
    $('ul').before('<p>hello jQuery</p>');
}
function f4(){
    // $('ul li:eq(1)').remove();
    $('ul li:eq(1)').empty();
}
function f5(){
    $('ul li').remove('#12');
}
</script>
</head>
<body>
<!-- DOM 操作：创建、插入、删除 -->
<!--
    创建：$(html)
    插入节点：
        append ( )：作为最后一个孩子添加进来。
        prepend ( )：作为第一个孩子添加进来。
        after ( )：作为下一个兄弟添加进来
        before ( )：作为上一个兄弟添加进来
    删除节点：
        remove ( )
        remove ( selector )
        empty ( )：清空节点
-->
<a href="javascript:;"
    onclick="f2();" >DIV 的内容在哪里?</a>
<ul>
```

```
<li>item1</li>
<li id="l2">item2</li>
<li>item3</li>
</ul>
<input type="button" value="插入"
  onclick="f3();" />
<input type="button" value="删除"
  onclick="f5();" />
</body>
</html>
```

d03.html 文件代码如下：

```
<html>
<head>
<meta http-equiv="Content-Type"
  content="text/html; charset=UTF-8">
<title>Insert title here</title>
<script type="text/javascript" src="../../js/j03.js">
</script>
</head>
<body>
  <div id="d1">hello jQuery</div>
</body>
</html>
```

j03.js 文件代码如下：

```
window.onload = function(){
  var obj = document.getElementById('d1');
  obj.onclick=function(){
    this.innerHTML = 'hello Ajax';
  };
};
```

d04.html 文件代码如下：

```
<html>
<head>
  <meta http-equiv="Content-Type"
    content="text/html; charset=UTF-8">
  <title>Insert title here</title>
  <script type="text/javascript"
    src="../../js/jquery-1.11.1.js"></script>
  <script type="text/javascript"
    src="../../js/j04.js"></script>
</head>
<body >
  <div id="d1">hello jQuery</div>
</body>
</html>
```

j04.文件代码如下：

```
$(function(){
  $('#d1').click(function(){
    $(this).html('hello Ajax');
  });
});
```


d05.html 文件代码如下：

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Insert title here</title>
  <script type="text/javascript" src="../js/jquery-1.11.1.js"></script>
  <script type="text/javascript">
    $(function(){
      $('ul li:eq(2)').click(function(){
        $(this).css('font-size','30px');
      });

      $('#a1').click(function(){
        var $obj = $('ul li:eq(2)').clone(true);
        $('ul').append($obj);
      });
    });
  </script>
</head>
<body >
  <!-- 复制节点 -->
  <!--
    clone ( ):
    clone ( true ): 使复制的节点也具有行为
                    ( 将处理代码一块复制 )
  -->
  <ul>
    <li>item1</li>
    <li>item2</li>
    <li>item3</li>
  </ul>
  <a href="javascript:;" id="a1">复制节点</a>
</body>
</html>
```

d06.html 文件代码如下：

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Insert title here</title>
  <style>
    .s1{
      color:red;
    }
    .s2{
      font-size:20px;
    }
  </style>
  <script type="text/javascript" src="../js/jquery-1.11.1.js"></script>
  <script type="text/javascript">
    $(function(){
      $('#a1').click(function(){
        //$('#d1').attr('style','color:red;font-size:30px;');
        //$('#d1').attr('class','s1');
        //$('#d1').addClass('s1');
        //$('#d1').addClass('s1 s2');
        //$('#d1').removeClass('s1');
        //$('#d1').toggleClass('s2');
        //alert($('#d1').hasClass('s1'));
        $('#d1').css('font-style','italic');
      });
    });
  </script>
</body>
</html>
```

```

    });
  });
</script>
</head>
<body>
  <!-- 样式操作 -->
  <!--
    获取和设置: attr("class" , " ") 或者 attr("style", " ");
    追加: addClass(' ')
    移除: removeClass(' ')或者
           removeClass('s1 s2')或者
           removeClass( ) 会删除所有样式
    切换样式: toggleClass, 有该样式就删除, 没有就添加
    是否有某个样式 hasClass(' ')
    读取 css(' ')
    设置 css ( ' ', ' ' )
    或者 css({ ' ': ' ', ' ': ' ' })设置多个样式
  -->
  <div id="d1">hello jQuery</div>
  <a href="javascript:;" id="a1">样式操作</a>
</body>
</html>

```

d07.html 文件代码如下：

```

<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Insert title here</title>
  <script type="text/javascript" src="../js/jquery-1.11.1.js"></script>
  <script type="text/javascript">
    $(function(){
      $('#a1').click(function(){
        var $obj = $('#d1').children('div');
        alert($obj.length);
        $obj.css('font-size','60px');

        $('#d3').next().css('font-size','60px');
        $('#d3').next('p').css('font-size','60px');

        $('#d3').prev().css('font-size','60px');

        $('#d3').siblings().css('font-size','60px');
        $('#d3').siblings('div').css('font-size','60px');

        $('#d1').find('p').css('font-size','60px');

        alert($('#d3').parent().attr('id'));
      });
    });
  </script>
</head>
<body>
  <!-- 遍历节点 -->
  <!--
    children( ) / children(selector) :
      只考虑子元素，不考虑其它后代
    next( ) / next ( selector ) : 下一个兄弟
  -->

```

```
prev ( ) / prev(selector) : 上一个兄弟
siblings ( ) / siblings(selector) : 其它兄弟
find ( selector ) : 查找满足选择器的所有后代
parent ( ) : 父节点 ( 没有选择器 )
-->
<div id="d1">
  <div id="d2">hello 1</div>
  <div id="d3">hello 2</div>
  <div id="d4">hello 3</div>
  <p>hello 4</p>
</div>
<a href="javascript:;" id="a1">遍历节点</a>
</body>
</html>
```

课后作业

1. 下列说法正确的是（ ）

- A . 使用 jQuery 时无需引入任何文件
- B . 进行 jQuery 操作时，第一步是使用选择器定位元素
- C . 使用 “\$” 符号封装的对象和 document.getElementById(“XX”)获取的对象是等价的
- D . 使用\$('#d1')可以定位到所有 id 属性是 d1 的文档元素

2. 编写程序，实现如图所示的页面效果

选中其中一列的复选框时，该复选框所在行的背景色高亮显示（黄色）。取消选中复选框时，所在行的背景色恢复。界面显示效果如图-1 所示：

员工信息			
	姓名	薪水	年龄
<input type="checkbox"/>	张三	20000	23
<input checked="" type="checkbox"/>	李四	22000	22
<input type="checkbox"/>	王五	14000	26
<input type="checkbox"/>	马六	15000	21

图 - 1

AJAX 和 jQuery

Unit04

知识体系.....Page 128

jQuery 事件处理	事件处理	使用jQuery 实现事件绑定
		获得事件对象 event
		事件对象的常用属性
	事件冒泡	什么是事件冒泡
		如何取消事件冒泡
	合成事件	jQuery 的合成事件种类
jQuery 增强操作	模拟操作	模拟操作的语法
	jQuery 动画	显示、隐藏的动画效果
		上下滑动式的动画实现
		淡入淡出式动画效果
		自定义动画效果
	jQuery 类数组	什么是类数组
		类数组的操作
	jQuery 对 AJAX 的支持	load ()
		get ()
		ajax ()

经典案例.....Page 134

jQuery 处理事件	使用jQuery 实现事件绑定
	获得事件对象 event
	事件对象的常用属性
事件冒泡	如何取消事件冒泡
合成事件	jQuery 的合成事件种类
模拟操作	模拟操作的语法
jQuery 动画	显示、隐藏的动画效果
	上下滑动式动画效果
	淡入淡出式动画效果
	自定义动画效果
jQuery 遍历操作	类数组的操作

jQuery 对 Ajax 的支持	load ()
	get ()
	ajax ()

课后作业.....Page 163

1. jQuery 事件处理

1.1. 事件处理

1.1.1. 【事件处理】使用 jQuery 实现事件绑定

使用jQuery实现事件绑定

Tarena
达内科技

- 语法：
 - `$obj . bind(事件类型, 事件处理函数)`
 - 如：`$obj . bind('click' , fn);`
 - 简写形式 `$obj . click (fn);`
- 注：`$obj.click()` 则代表触发了click事件。

+

1.1.2. 【事件处理】获得事件对象 event

获得事件对象event

Tarena
达内科技

- 只需要为事件处理函数传递任意一个参数
- 如：`$obj.click(function(e) { ... })`
- e就是事件对象，但已经经过jQuery对底层事件对象的封装
- 封装后的事件对象可以方便的兼容各浏览器

+

1.1.3. 【事件处理】事件对象的常用属性

事件对象的常用属性

Tarena
达内科技

- 获取事件源 `var obj = e.target` 返回值是DOM对象
- 获取鼠标点击的坐标
 - `e.pageX`
 - `e.pageY`

+

1.2. 事件冒泡

1.2.1. 【事件冒泡】什么是事件冒泡

Tarena
达内科技

什么是事件冒泡

- 子节点产生的事件会依次向上抛给父节点

```

graph BT
    A((a)) -- "1. 发生点击" --> E1[3. 事件对象]
    A --> D((div))
    D -- "2. 事件对象" --> E2[3. 事件对象]
    D --> B((body))
    E1 --> H1[绑定了事件处理函数]
    E2 --> H2[绑定了事件处理函数]
    
```

+

1.2.2. 【事件冒泡】如何取消事件冒泡

Tarena
达内科技

如何取消事件冒泡

- `e.stopPropagation()` 可以取消事件冒泡
- 如：

```

$( 'div' ).click(function(e){
    alert( '点击了div' );
});
$( 'a' ).click(function ( e ){
    alert( '点击了一个链接' );
    e.stopPropagation( );
});
    
```

+

1.3. 合成事件

1.3.1. 【合成事件】jQuery 的合成事件种类

Tarena
达内科技

jQuery的合成事件种类

- `hover (mouseenter , mouseleave)` 模拟光标悬停事件
- `toggle ()` 在多个事件响应中切换

+

1.4. 模拟操作

1.4.1. 【模拟操作】模拟操作的语法

<div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div>	<div><div>模拟操作的语法</div><div><div>Tarena 达内科技</div></div><ul style="list-style-type: none">• <code>\$obj.trigger(事件类型)</code>• 如：<code>\$obj.trigger("focus");</code>• 简写形式 <code>\$obj . focus();</code><div>知识讲解</div><div>+</div></div>
---	---

2. jQuery 增强操作

2.1. jQuery 动画

2.1.1. 【jQuery 动画】显示、隐藏的动画效果

<div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div>	<div><div>显示、隐藏的动画效果</div><div><div>Tarena 达内科技</div></div><ul style="list-style-type: none">• <code>show () / hide ()</code>• 作用：通过同时改变元素的宽度和高度来实现显示或者隐藏• 用法：<code>\$obj . show (执行时间, 回调函数);</code> 执行时间：<code>slow</code> , <code>normal</code> , <code>fast</code>或毫秒数 回调函数：动画执行完毕之后要执行的函数<div>知识讲解</div><div>+</div></div>
---	--

2.1.2. 【jQuery 动画】上下滑动式的动画实现

<div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div>	<div><div>上下滑动式的动画实现</div><div><div>Tarena 达内科技</div></div><ul style="list-style-type: none">• <code>slideDown () / slideUp ()</code>• 作用：通过改变高度来实现显示或者隐藏的效果• 用法同 <code>show () / hide ()</code><div>知识讲解</div><div>+</div></div>
---	--

2.1.3. 【jQuery 动画】淡入淡出式动画效果

jQuery动画特效

Tarena
达内科技


淡入淡出式动画效果

- `fadeIn () / fadeOut ()`
- 作用：通过改变不透明度opacity来实现显示或者隐藏
- 用法同 `show / hide`

知识讲解

+

2.1.4. 【jQuery 动画】自定义动画效果



自定义动画效果

- animate ()
- 用法：animate (js对象, 执行时间, 回调函数) ;
- js对象：{ }描述动画执行之后元素的样式
- 执行时间：毫秒数
- 回调函数：动画执行结束后要执行的函数
- 如：



```
$( "div" ). click (function(){
    $(this).animate({ 'left' : ' 500px' },4000);
    $(this).animate({ 'top' : ' 300px' },2000)
        .fadeOut( 'slow' );
});
```



2.2. jQuery 类数组

2.2.1. 【jQuery 类数组】什么是类数组

[illegible]



2.2.2. 【jQuery 类数组】类数组的操作

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3>类数组的操作</h3> <ul style="list-style-type: none"> length 属性 each (fn) 遍历类数组，fn 用来处理 DOM 对象。在 fn 中 this 表示正在被遍历的那个 DOM 对象。fn 函数可以添加一个参数 i 用于表示正在被遍历的 DOM 对象的下标（从 0 开始） eq(index)：将下标等于 index 的 DOM 对象取出来 get ()：返回一个 DOM 对象组成的数组 index (obj)：返回 DOM 或 jQuery 对象在类数组中的下标 <div style="text-align: right;">  </div>
---	---



<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3>类数组的操作（续）</h3> <pre> item1 item2 item3 </pre> <pre>var \$obj = \$('ul li'); \$obj.each(function(){ if(i==0) \$(this).css('font-size' , ' 30px'); });</pre> <div style="text-align: right;">  </div>
---	--

2.3. jQuery 对 AJAX 的支持



2.3.1. 【jQuery 对 AJAX 的支持】load ()

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3>load ()</h3> <ul style="list-style-type: none"> 作用：将服务器返回的数据字节添加到符合要求的节点之上 用法： \$obj.load (请求地址，请求参数)； 请求参数 <ul style="list-style-type: none"> “username=tom & age=22” { 'username' : 'tom' , 'age' : 22 } 有请求参数时，load 方法发送 POST 请求，否则发送 GET 请求 <div style="text-align: right;">  </div>
---	--

2.3.2. 【jQuery 对 AJAX 的支持】get ()

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>get ()</h4> <ul style="list-style-type: none"> • 作用：发送GET类型的请求 • 用法：\$.get(请求地址, 请求参数, 回调函数, 服务器返回的数据类型) <p>说明：</p> <ul style="list-style-type: none"> - 回调函数添加的参数是服务器返回的数据 - 服务器返回的数据类型： <ul style="list-style-type: none"> html：HTML文本 text：文本 JSON：js对象 xml：XML文档 script：JavaScript脚本 <div style="text-align: right;">  </div>
---	---

2.3.3. 【jQuery 对 AJAX 的支持】ajax ()

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>ajax ()</h4> <ul style="list-style-type: none"> • 用法：\$.ajax({}); • ()内可以设置的选项参数： <ul style="list-style-type: none"> - url：请求地址 - type：请求方式 - data：请求参数 - dataType：服务器返回的数据类型 - success：服务器处理正常对应的回调函数 - error：服务器出错对应的回调函数 - async：true (缺省)，当值为false时发送同步请求 <div style="text-align: right;">  </div>
---	---

经典案例

1. jQuery 处理事件

- 问题

如何使用 jQuery 进行事件处理。

- 方案

通过 jQuery 提供的与事件有关的 API 进行事件绑定、获取事件源、完成合成事件等操作。

- 步骤

步骤一：事件绑定及获取事件源

新建 e1.html 页面：

```
<body>
  <!-- 绑定事件处理函数 -->
  <!--
    $obj.bind(事件类型, 事件处理函数);
    比如: $obj.bind('click' , fn );
  -->
  <div id="d1">hello jQuery</div>
  <a href="javascript:;" id="a1">绑定事件处理函数</a>
</body>
```

图 - 1

添加事件：

```
<script type="text/javascript">
  $(function() {
    $('#a1').bind('click',function(){
      $('#d1').html('hello java');
    });
  });
</script>
```

图 - 2

运行结果：

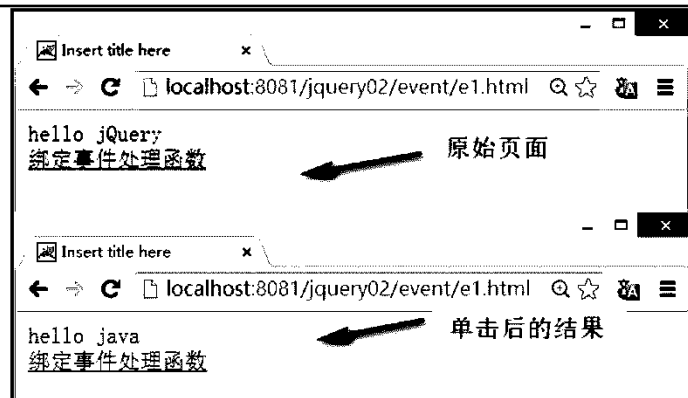


图 - 3

获取事件对象：

新建 e2.html 页面：

```
<body>
  <!-- 获取事件对象 -->
  <a href="javascript:;">第一个链接</a><br/>
  <a href="javascript:;">第二个链接</a>
</body>
```

图 - 4

添加事件：

```
<script type="text/javascript">
$(function() {
  $('a').click(function(e) {
    //通过事件对象找到事件源
    var obj = e.target; //返回值是一个dom对象
    alert(obj.innerHTML);

    //获得鼠标点击的坐标
    alert(e.pageX + ' ' + e.pageY);
  });
});
```

图 - 5

运行结果：

点击第一个链接后的结果：

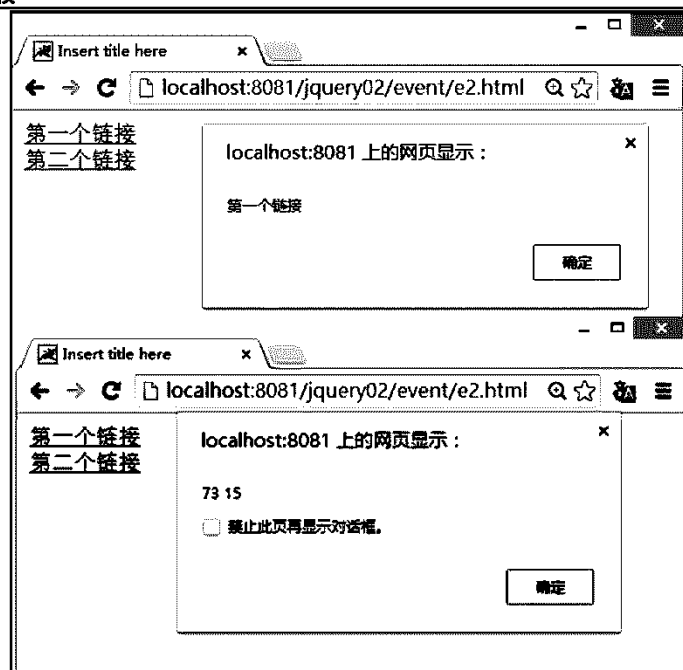


图 - 6

点击第二个链接后的结果：

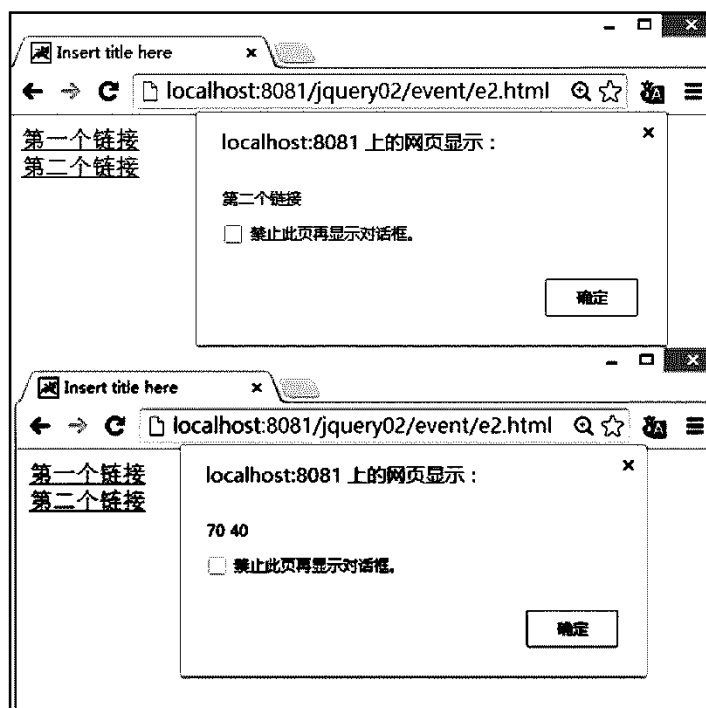


图 - 7

• 完整代码

e1.html 文件代码如下：

```
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
<script type="text/jav
ascript" src="../../js/jquery-1.11.1.js"></script>
<script type="text/javascript">
    $(function(){
        $('#a1').bind('click',function(){
            $('#d1').html('hello java');
        });
    });
</script>
</head>
<body>
    <!-- 绑定事件处理函数 -->
    <!--
        $obj.bind(事件类型, 事件处理函数);
        比如: $obj.bind('click' , fn );
    -->
    <div id="d1">hello jQuery</div>
    <a href="javascript:;" id="a1">绑定事件处理函数</a>
</body>
</html>
```

e2.html 文件代码如下：

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Insert title here</title>
    <script type="text/javascript" src="../../js/jquery-1.11.1.js"></script>
    <script type="text/javascript">
        $(function(){
            $('a').click(function(e){
                //通过事件对象找到事件源

                var obj = e.target; //返回值是一个 dom 对象
                alert(obj.innerHTML);

                //获得鼠标点击的坐标
                alert(e.pageX + ' ' + e.pageY);
            });
        });
    </script>
</head>
<body>
    <!-- 获取事件对象 -->
    <a href="javascript:;">第一个链接</a><br/>
    <a href="javascript:;">第二个链接</a>
</body>
</html>
```

2. 事件冒泡

• 问题

取消子节点向上抛出事件。

- 方案

使用事件的 `stopPropagation()` 方法。

- 步骤

新建 `e3.html` 文件：

```
<body>
  <!-- 事件冒泡 -->
  <!--
    什么是事件冒泡?子节点产生的事件会依次向上抛给父节点
  -->
  <div style="width:200px;height:200px;
border:1px solid red;">
    <a href="javascript:;">hello jQuery</a>
  </div>
</body>
```

图 - 8

添加事件：

```
<script type="text/javascript">
  $(function() {
    $('div').click(function(e) {
      alert('你点击了一个div');
    });

    $('a').click(function(e) {
      alert('你点击了一个链接');
      //取消事件冒泡
      e.stopPropagation();
    });
  });
</script>
```

图 - 9

运行结果：

如果不添加 `e.stopPropagation()` 方法，则运行结果如下所示：



图 - 10

添加 e.stopPropagation()方法后，运行结果如下所示：

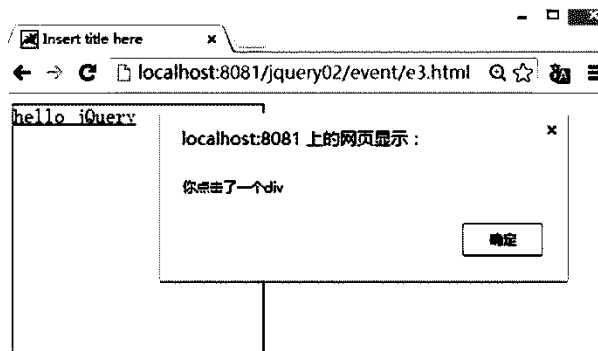


图 - 11

• 完整代码

e3.html 文件代码如下：

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Insert title here</title>
  <script type="text/javascript" src="../../js/jquery-1.11.1.js"></script>
  <script type="text/javascript">
    $(function(){
      $('div').click(function(e){
        alert('你点击了一个div');
      });

      $('a').click(function(e){
        alert('你点击了一个链接');
        //取消事件冒泡
        e.stopPropagation();
      });
    });
  </script>
</head>
<body>
  <div>
    <a href="#">hello jQuery</a>
  </div>
</body>
</html>
```

```

    });
  });
</script>
</head>
<body>
  <!-- 事件冒泡 -->
  <!-- 什么是事件冒泡?子节点产生的事件会依次向上抛给父节点 -->
  <div style="width:200px;height:200px;border:1px solid red;">
    <a href="javascript:;">hello jQuery</a>
  </div>
</body>
</html>

```

3. 合成事件

- 问题

多个事件状态的组合事件。

- 方案

使用 hover()、toggle()方法实现合成事件。

- 步骤

步骤一：新建 e4.html 文件：

```

<!-- 合成事件 :hover (mouseenter , mouseleave)
      模拟光标悬停事件
-->
<div class="s1">
</div>

```

图 - 12

添加事件：

```

$(function() {
  $( '.s1' ).mouseenter(function() {
    $(this).addClass('s2');
  });
  $( '.s1' ).mouseleave(function() {
    $(this).removeClass('s2');
  });
  // 等价
  $( '.s1' ).hover(function() {
    $(this).addClass('s2');
  },function() {
    $(this).removeClass('s2');
  });
});

```

图 - 13

运行结果：



图 - 14

步骤二：模拟连续单击动作

新建 e5.html 文件：

```
<body>
  <!-- toggle模拟连续单击事件 -->
  <a href="javascript:;">连续单击</a>
  <div style="display:none;">隐藏的Div</div>
</body>
```

图 - 15

添加事件：

```
$(function(){
    $('a').click(function(){
        $('div').toggle();
    });
});
```

图 - 16

查看运行结果：随着点击超链接，会出现 div 交替的显示和隐藏。

• 完整代码

e4.html 文件代码如下：

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Insert title here</title>
  <style>
    .s1{
      width:100px;
```

```

        height:100px;
        border:1px solid red;
    }
    .s2{
        border:10px solid red;;
    }
</style>
<script type="text/javascript" src="../../js/jquery-1.11.1.js"></script>
<script type="text/javascript">
    $(function(){
        /*
            $('.s1').mouseenter(function(){
                $(this).addClass('s2');
            });

            $('.s1').mouseleave(function(){
                $(this).removeClass('s2');
            });
        */

        $('.s1').hover(function(){
            $(this).addClass('s2');
        },function(){
            $(this).removeClass('s2');
        });
    });
</script>
</head>
<body>
    <!-- 合成事件 :hover(mouseenter , mouseleave)
        模拟光标悬停事件
    -->
    <div class="s1">
    </div>
</body>
</html>

```

e5.html 文件代码如下：

```

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
<script type="text/javascript" src="../../js/jquery-1.11.1.js"></script>
<script type="text/javascript">
    $(function(){
        $('a').toggle(function(){
            //$('#div').show('slow');
            $(this).html('修改后的 HTML 内容');
        },function(){
            //$('#div').hide('fast');
            $(this).html('第二部分内容');
        });
    });
</script>
</head>
<body>
    <!-- toggle (fn1 , fn2 , fn3 ..) 模拟连续单击事件 -->
    <a href="javascript:;">连续单击</a>
    <div style="display:none;">隐藏的 Div</div>
</body>
</html>

```

4. 模拟操作

- 问题

模拟页面元素发生得到焦点事件。

- 方案

使用 `trigger()` 等方法实现。

- 步骤

步骤一：新建 `e6.html` 文件：

```
<body>
  <!-- 模拟操作 -->
  <!--
    a, 正式写法 $obj.trigger(事件类型)
      如 $obj.trigger("focus");
    b, 简写形式 $obj.focus( ) ;
  -->
  username:<input name="username"
  id="username"/><br/>
  name:<input name="name"/>
  <input type="button" value="模拟操作" id="b1"/>
</body>
```

图 - 17

添加事件：

```
$(function() {
  $('#b1').click(function() {
    //$('#username').trigger('focus');
    $('#username').focus();
  });
});
```

图 - 18

运行结果：

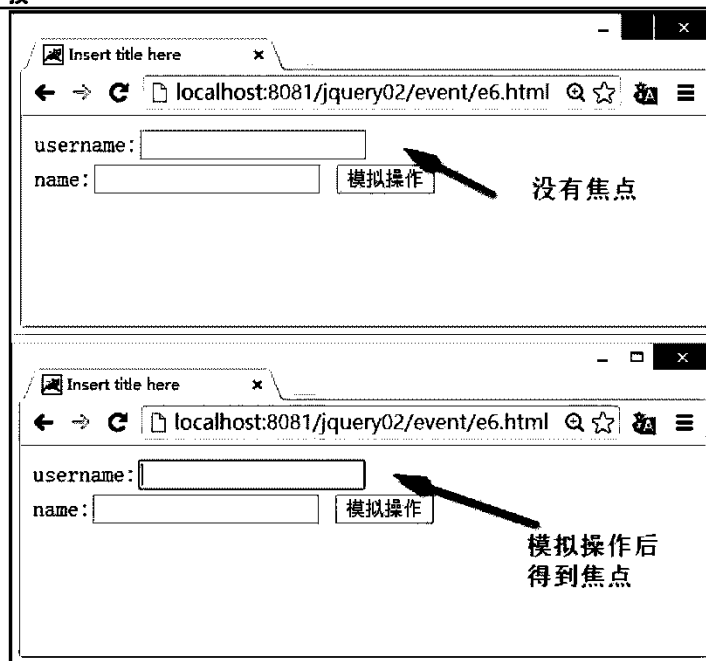


图 - 19

• 完整代码

e6.html 文件代码如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
<script type="text/javascript" src="../js/jquery-1.11.1.js"></script>
<script type="text/javascript">
    ${function(){
        $('#b1').click(function(){
            //$('#username').trigger('focus');
            $('#username').focus();
        });
    }}
</script>
</head>
<body>
<!-- 模拟操作 -->
<!--
    a. 正式写法  $obj.trigger(事件类型)
        如 $obj.trigger("focus");
    b. 简写形式  $obj.focus( ) ;
-->
username:<input name="username"
id="username"/><br/>
name:<input name="name"/>
<input type="button" value="模拟操作" id="b1"/>
</body>
</html>
```

5. jQuery 动画

- 问题

jQuery 中的动画函数介绍。

- 方案

使用 jQuery 中与动画有关的 API 实现显示、隐藏、淡入、淡出、自定义动画的效果。

- 步骤

步骤一：显示/隐藏

新建 a1.html 文件：

```
<!-- show () / hide () -->
<!--
    a, 作用：通过同时改变元素的宽度和高度来实现显示或者隐藏。
    b, 用法：$obj.show ( 执行时间 , 回调函数 ) ;
    执行时间：'slow' 'normal' 'fast' 或者用毫秒数
    回调函数：当动画执行完毕之后执行的函数。
-->
<!-- slideDown () / slideUp () -->
<!--
    a, 作用：通过改变高度来实现显示或者隐藏的效果。
    b, 用法：同 $obj.show()。
-->
<input type="button" id="btn1" value="显示DIV"/>
<input type="button" id="btn2" value="隐藏DIV"/>
<div>显示或隐藏的内容。</div>
```

图 - 20

添加事件：

```
<script type="text/javascript">
    $(function() {
        $('#btn1').click(function() {
            $('#div').show(3000);
            //$('#div').slideDown(3000);
        });
        $('#btn2').click(function() {
            $('#div').hide(
                'normal',
                function() {alert('消失了');}
            );
            //$('#div').slideUp(3000);
        });
    });
</script>
```

图 - 21

显示效果：

页面加载完毕，DIV 是显示的。点击 btn2 之后，DIV 消失，并在消失后弹出消息框，显示文本。点击 btn1 之后，DIV 显示。

添加事件：

```
$(function() {
    $('#btn1').click(function() {
        //$('#div').show(3000);
        $('#div').slideDown(3000);
    });
    $('#btn2').click(function() {
        //$('#div').hide(
        //    'normal',
        //    function() {alert('消失了');}
        //);
        $('#div').slideUp(3000);
    });
});
```

图 - 22

显示效果：

DIV 显示时，以自上而下的方式呈现。DIV 隐藏时，以自下而上的方式消失。

步骤二：淡入/淡出

新建 a2.html 文件：

```
<body>
    <!-- fadeIn () 淡入 / fadeOut () 淡出 -->
    <!-- 作用：通过改变不透明度opacity来实现显示或者隐藏 -->
    <input type="button" id="btn1" value="淡入"/>
    <input type="button" id="btn2" value="淡出"/>
    <div id="d1">淡入或淡出的内容。</div>
</body>
```

图 - 23

添加事件：

```
<script type="text/javascript">
    $(function() {
        $('#btn1').click(function() {
            $('#div').fadeIn(3000);
        });
        $('#btn2').click(function() {
            $('#div').fadeOut();
        });
    });
</script>
```

图 - 24

运行效果：div 会以改变透明度的方式，变浅或变深的显示和隐藏。

步骤三：自定义动画

新建 a3.html 文件：

```
<body>
  <!-- animate (js对象, 执行时间, 回调函数);
        js对象: { } 描述动画执行之后, 元素的样式。
        比如 $obj.animate({'left': '200px'})
  -->
  <div id="d1"></div>
</body>
```

图 - 25

添加事件：

```
$(function() {
    $('#d1').click(function() {
        //1. 直着走
        $(this).animate({'left': '500px'}, 2000);
    });
});
```

图 - 26

```
$(function() {
    $('#d1').click(function() {
        //2. 斜着走
        $(this).animate(
            {'left': '500px', 'top': '200px'}, 2000);
    });
});
```

图 - 27

```
$(function() {
    $('#d1').click(function() {
        //3. 先直着走, 再斜着走 (动画队列)
        $(this).animate({'left': '500px'}, 2000);
        $(this).animate({'top': '200px'}, 2000);
    });
});
```

图 - 28

```
$(function() {
    $('#d1').click(function() {
        //4. 走完后消失
        $(this).animate(
            {'left': '500px', 'top': '300px'}, 2000)
            .fadeOut('slow');
    });
});
```

图 - 29

运行效果：DIV 在页面中沿着坐标指定的方向移动。图-26 的路线为向右平移，图-27 的路线为向右下方平移，图-28 的路线为向右平移后向下平移，图-29 的路线为向右下方平移后消失。

• 完整代码

a1.html 文件代码如下：

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Insert title here</title>
  <style>
    div{
      width:450px;
      height:40px;
      background-color:#fff8dc;
      border:1px solid black;
      display:block;
    }
  </style>
  <script type="text/javascript" src="../js/jquery-1.11.1.js"></script>
  <script type="text/javascript">
    $(function(){
      $('#btn1').click(function(){
        //$('#div').show(3000);
        $('#div').slideDown(3000);
      });
      $('#btn2').click(function(){
        //$('#div').hide(
        //  'normal',
        //  function(){alert('消失了');}
        //);
        $('#div').slideUp(3000);
      });
    });
  </script>
</head>
<body>
  <!-- show ( ) / hide ( ) -->
  <!--
    a , 作用：通过同时改变元素的宽度和高度来实现显示或者隐藏。
    b , 用法：$obj.show ( 执行时间 , 回调函数 ) ;
    执行时间：'slow' 'normal' 'fast' 或者用毫秒数
    回调函数：当动画执行完毕之后执行的函数。
  -->
  <!-- slideDown ( ) / slideUp ( ) -->
  <!--
    a , 作用：通过改变高度来实现显示或者隐藏的效果。
    b , 用法：同 $obj.show()。
  -->
  <input type="button" id="btn1" value="显示 DIV"/>
  <input type="button" id="btn2" value="隐藏 DIV"/>
  <div>显示或隐藏的内容。</div>
</body>
</html>
```

a2.html 文件代码如下：

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Insert title here</title>
  <style>
    #d1{
      width:200px;
      height:200px;
      border-color:red;
      display:none;
    }
  </style>
  <script type="text/javascript" src="../js/jquery-1.11.1.js"></script>
  <script type="text/javascript">
    $(function(){
      $('#btn1').click(function(){
        $('#div').fadeIn(3000);
      });
      $('#btn2').click(function(){
        $('#div').fadeOut();
      });
    });
  </script>
</head>
<body>
  <!-- fadeIn ( ) 淡入 / fadeOut ( ) 淡出 -->
  <!-- 作用：通过改变不透明度 opacity 来实现显示或者隐藏 -->
  <input type="button" id="btn1" value="淡入"/>
  <input type="button" id="btn2" value="淡出"/>
  <div id="d1">淡入或淡出的内容。</div>
</body>
</html>
```

a3.html 文件代码如下：

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Insert title here</title>
  <style>
    #d1{
      width:100px;
      height:100px;
      background-color:red;
      left:100px;
      top:80px;
      position:absolute;
    }
  </style>
  <script type="text/javascript" src="../js/jquery-1.11.1.js"></script>
  <script type="text/javascript">
    $(function(){
      $('#d1').click(function(){
        //1.直着走
        //$(this).animate({'left':'500px'},2000);
        //2.斜着走
        //$(this).animate(
        //  {'left':'500px','top':'200px'},2000);
      });
    });
  </script>
</body>
</html>
```

```

//3.先直着走,再斜着走(动画队列)
//$(this).animate({'left':'500px'},2000);
//$(this).animate({'top':'200px'},2000);
//4.走完后消失
$(this).animate(
    {'left':'500px','top':'300px'},2000)
    .fadeOut('slow');
    });
</script>
</head>
<body>
    <!-- animate(js 对象,执行时间,回调函数);
        js 对象: { } 描述动画执行之后,元素的样式。
        比如 $obj.animate({'left':'200px'}
    -->
    <div id="d1"></div>
</body>
</html>

```

6. jQuery 遍历操作

- 问题

使用 jQuery 实现对数组的遍历操作。

- 方案

使用 jQuery 的 length 属性、each()、eq()、get()、index()方法对类似数组的元素进行操作。

- 步骤

步骤一：新建 array/a1.html 文件

```

<body>
    <!-- 类数组操作 -->
    <ul>
        <li>item1</li>
        <li>item2</li>
        <li>item3</li>
    </ul>
    <input type="button" value="类数组操作" id="b1"/>
</body>

```

图 - 30

步骤二：使用 length 属性获取个数

```
$(function(){
    $('#b1').click(function(){
        var $obj = $('ul li');
        alert($obj.length);
    });
});
```

图 - 31

步骤三：使用 each (fn) 遍历

```
$(function(){
    $('#b1').click(function(){
        var $obj = $('ul li');
        $obj.each(function(i){
            //i:正在被遍历的dom对象的下标
            if(i == 1){
                //this:正在被遍历的dom对象
                $(this).css('font-size','80px');
            }
        });
    });
});
```

图 - 32

步骤四：使用 eq (index) 取 jQuery 对象

```
$(function(){
    $('#b1').click(function(){
        var $obj = $('ul li');
        var $o = $obj.eq(1);
        $o.css('font-size','60px');
    });
});
```

图 - 33

步骤五：使用 get (index) 取 DOM 对象或 DOM 数组

```
$(function(){
    $('#b1').click(function(){
        var $obj = $('ul li');
        var o = $obj.get(1); DOM对象
        o.innerHTML = 'hello jQuery';

        var $obj = $('ul li');
        var arr = $obj.get(); DOM数组
        alert(arr[1].innerHTML);
    });
});
```

图 - 34

步骤六：使用 index (obj) 获取 DOM 或 jQuery 的下标

```
$(function(){
    $('#b1').click(function(){
        var $obj = $('ul li');
        var o = $obj.get(1);
        var index = $obj.index(o);
        alert(index);
    });
});
```

O对象的坐标

图 - 35

• 完整代码

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Insert title here</title>
    <script type="text/javascript" src="../js/jquery-1.11.1.js"></script>
    <script type="text/javascript">
        $(function(){
            $('#b1').click(function(){
                /*
                var $obj = $('ul li');
                //alert($obj.length);
                $obj.each(function(i){
                    //i:正在被遍历的 dom 对象的下标
                    if(i == 1){
                        //this:正在被遍历的 dom 对象
                        $(this).css('font-size','80px');
                    }
                });
                */

                //var $obj = $('ul li');
                //var $o = $obj.eq(1);
                //$.css('font-size','60px');

                //var o = $obj.get(1);
                //o.innerHTML = 'hello jQuery';

                //var $obj = $('ul li');
                //var o = $obj.get(1);
                //var index = $obj.index(o);
                //alert(index);

                var $obj = $('ul li');
                var arr = $obj.get();
                alert(arr[1].innerHTML);
            });
        });
    </script>
</head>
<body>
    <!-- 类数组操作 -->
    <!--
        a , length 属性：dom 对象的个数。
        b , each ( fn ) 用来遍历类数组。fn 用来处理 dom 对象。
    -->
```

在 fn 函数里面 this 表示正在被遍历的那个 dom 对象，

另外 fn 函数可以添加一个参数，

比如 i，表示正在被遍历的那个 dom 对象的下标（下标从 0 开始）

c,eq(index): 将下标等于 index 的 dom 对象取出来，并且封装成一个 jquery 对象。

d,get(index): 将下标等于 index 的 dom 对象取出来。

e,get(): 返回一个 dom 对象组成的数组。

f,index(obj): 返回 dom 或 jquery 对象在类数组中的下标

```
-->
<ul>
  <li>item1</li>
  <li>item2</li>
  <li>item3</li>
</ul>
<input type="button" value="类数组操作" id="b1"/>
</body>
</html>
```

7. jQuery 对 Ajax 的支持

• 问题

jQuery 对 Ajax 的实现。

• 方案

使用\$.load(),\$.get(),\$.post(),\$.ajax()方法发送异步请求。

• 步骤

步骤一：\$.load()方法

新建 emps.jsp 页面：

```
<table cellpadding="0" cellspacing="0"
  width="50%" border="1">
<tr><td>员工号</td><td>员工姓名</td><td>&nbsp;</td>
</tr>
<tr>
  <td>T10001</td><td>张三</td><td>
    <a href="javascript:;"class="s1">
      显示工资明细</a>
    <div></div>
  </td>
</tr>
<tr><td>T10002</td><td>李四</td><td>
  <a href="javascript:;"class="s1">
    显示工资明细</a>
  <div></div>
  </td>
</tr>
</table>
```

图 - 36

添加异步请求代码：

```
$(function(){
    $('.s1').click(function(){
        var eId =
            $(this).parent().siblings().eq(0).text();
            $(this).next().load('salary.do','eId=' + eId);
    });
});
```

图 - 37

新建 ActionServlet 类：

```
response.setContentType("text/html;charset=utf-8");
PrintWriter out = response.getWriter();
String uri = request.getRequestURI();
String action =
    uri.substring(uri.lastIndexOf("/"),uri.lastIndexOf("."));
if(action.equals("/salary")){
    String flight = request.getParameter("eId");
    if(flight.equals("T10001")){
        out.println("实际工资：¥10000<br/>个税：¥2500");
    }else{
        out.println("实际工资：¥20000<br/>个税：¥5000");
    }
}
```

图 - 38

运行查看结果：



The screenshot shows a web browser window with the address bar displaying 'localhost:8088/jquery02/emps.jsp'. The main content area contains a table with three columns: '员工号' (Employee ID), '员工姓名' (Employee Name), and a third column for salary details. The table has two data rows. The first row shows employee 'T10001' named '张三' (Zhang San) with salary details '实际工资：¥10000' (Actual Salary: ¥10000) and '个税：¥2500' (Income Tax: ¥2500). The second row shows employee 'T10002' named '李四' (Li Si) with salary details '实际工资：¥20000' (Actual Salary: ¥20000) and '个税：¥5000' (Income Tax: ¥5000). Each row has a link '显示工资明细' (Show Salary Details) above the salary information.

员工号	员工姓名	
T10001	张三	显示工资明细 实际工资：¥10000 个税：¥2500
T10002	李四	显示工资明细 实际工资：¥20000 个税：¥5000

图 - 39

步骤二：\$.get()/\$.post()方法

新建 stock.jsp 页面：

```
<div id="d1">
  <div id="d2">股票实时行情</div>
  <div id="d3">
    <table width="100%" cellpadding="0"
      cellspacing="0">
      <thead>
        <tr><td>代码</td><td>名称</td>
          <td>价格</td></tr>
      </thead>
      <tbody id="tb1">
      </tbody>
    </table>
  </div>
</div>
```

图 - 40

```
<style>
  #d1{
    width:450px;
    height:280px;
    background-color:black;
    margin-left:400px;
    margin-top:50px;
  }
  #d2{
    color:white;
    background-color:purple;
    height:35px;
  }
  table{
    color:white;
    font-size:24px;
  }
</style>
```

图 - 41

新建 Stock.java 文件：

```
package bean;

public class Stock {
    private String code;
    private String name;
    private double price;
    public String getCode() {
        return code;
    }
    public void setCode(String code) {
        this.code = code;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {
        this.price = price;
    }
}
```

图 - 42

修改 ActionServlet 类：

```
}else if(action.equals("/quote")){
    Random r = new Random();
    List<Stock> stocks =
        new ArrayList<Stock>();
    for(int i=0;i<8;i++){
        Stock s = new Stock();
        s.setCode("60001" + r.nextInt(10));
        s.setName("中国国贸" + r.nextInt(10));
        DecimalFormat df = new DecimalFormat("0.00");
        double price = Double.parseDouble(
            df.format(100 * r.nextDouble()));
        s.setPrice(price);
        stocks.add(s);
    }
    JSONArray jsonArr =
        JSONArray.fromObject(stocks);
    String jsonStr = jsonArr.toString();
    System.out.println(jsonStr);
    out.println(jsonStr);
}
```

图 - 43

发送 post 请求的代码：

```
$(function(){
    setInterval(quote,4000); //设置4秒的
    }); //间隔时间
function quote(){
    $.post('quote.do',function(data){
        //data:服务器返回的数据,如果
        //是json字符串,会自动转换成相应的
        //javascript对象。
        $('#tbl').empty();
        for(i=0;i<data.length;i++){
            var s = data[i];
            $('#tbl').append(
                '<tr><td>' + s.code
                + '</td><td>' + s.name
                + '</td><td>' + s.price
                + '</td></tr>');
        }
    },'json'); //返回数据格式的说明
}
```

图 - 44

运行查看结果：

股票实时行情		
代码	名称	价格
600015	中国国贸1	50.56
600015	中国国贸8	66.37
600017	中国国贸2	50.66
600015	中国国贸5	26.82
600010	中国国贸4	56.78
600018	中国国贸0	83.11
600010	中国国贸6	9.07
600019	中国国贸8	10.23

图 - 45

每隔四秒钟，信息就会更新一次，在不断的变化。

修改 post 请求为 get 请求时，相关参数不变，只有在 IE 浏览器下会存在缓存的问题，所以在请求地址后面追加一个随机数就可以欺骗 IE 浏览器，让其认为请求的数据有变化。运行结果同图-45 相同。

步骤三：\$.ajax()方法

新建 stock2.jsp 页面

基本结构和 CSS 样式和 stock.jsp 页面内容一致。

添加异步请求代码：

```
function quoto(){
    $.ajax({
        url:"quoto.do",
        type:"post",
        dataType:"json",
        success:function(data){
            $('#tbl1').empty();
            for(i=0;i<data.length;i++){
                var s = data[i];
                $('#tbl1').append(
                    '<tr><td>' + s.code
                    + '</td><td>' + s.name
                    + '</td><td>' + s.price
                    + '</td></tr>');
            }
        },
        error:function(){
            //服务器出错之后的处理逻辑。
        }
    });
}
```

图 - 46

运行查看结果，效果与图-45 所示结果相同。

• 完整代码

emps.jsp 文件代码如下：

```
<%@page pageEncoding="utf-8"
contentType="text/html; charset=utf-8" %>
<html>
<head>
    <script type="text/javascript" src="js/jquery-1.11.1.js"></script>
    <script type="text/javascript">
        $(function(){
            $('s1').click(function(){
                var eId =
                    $(this).parent().siblings().eq(0).text();
                    $(this).next().load('salary.do', 'eId=' + eId);
            });
        });
    </script>
</head>
<body>
    <table cellpadding="0" cellspacing="0"
        width="50%" border="1">
        <tr><td>员工号</td><td>员工姓名</td><td>&nbsp;</td>
        </tr>
        <tr>
            <td>T10001</td><td>张三</td><td>
                <a href="javascript:;" class="s1">
                    显示工资明细</a>
                <div></div>
            </td>
        </tr>
        <tr><td>T10002</td><td>李四</td><td>
                <a href="javascript:;" class="s1">
```

```

                显示工资明细</a>
            <div></div>
        </td>
    </tr>
</table>
</body>
</html>

```

stock.jsp 文件代码如下:

```

<%@page pageEncoding="utf-8"
contentType="text/html; charset=utf-8" %>
<html>
<head>
<style>
    #d1{
        width:450px;
        height:280px;
        background-color:black;
        margin-left:400px;
        margin-top:50px;
    }
    #d2{
        color:white;
        background-color:purple;
        height:35px;
    }
    table{
        color:white;
        font-size:24px;
    }
</style>
<script type="text/javascript" src="js/jquery-1.11.1.js"></script>
<script type="text/javascript">
    $(function(){
        setInterval(quote,4000);
    });
    function quote(){
        $.post('quote.do',function(data){
            //data:服务器返回的数据,如果
            //是 json 字符串,会自动转换成相应的
            //javascript 对象。
            $('#tbl1').empty();
            for(i=0;i<data.length;i++){
                var s = data[i];
                $('#tbl1').append(
                    '<tr><td>' + s.code
                    + '</td><td>' + s.name
                    + '</td><td>' + s.price
                    + '</td></tr>');
            }
        },'json');
    }
</script>
</head>
<body style="font-size:30px;font-style:italic;">
    <div id="d1">
        <div id="d2">股票实时行情</div>
        <div id="d3">
            <table width="100%" cellpadding="0"
            cellspacing="0">
                <thead>
                    <tr><td>代码</td><td>名称</td>

```

```
        <td>价格</td></tr>
    </thead>
    <tbody id="tbl1">
    </tbody>
</table>
</div>
</body>
</html>
```

stock2.jsp 文件代码如下：

```
<%@page pageEncoding="utf-8"
contentType="text/html; charset=utf-8" %>
<html>
<head>
    <style>
        #d1{
            width:450px;
            height:280px;
            background-color:black;
            margin-left:400px;
            margin-top:50px;
        }
        #d2{
            color:white;
            background-color:red;
            height:35px;
        }
        table{
            color:white;
            font-size:24px;
        }
    </style>
    <script type="text/javascript" src="js/jquery-1.11.1.js"></script>
    <script type="text/javascript">
        $(function(){
            setInterval(quote,4000);
        });
        function quote(){
            $.ajax({
                url:"quote.do",
                type:"post",
                dataType:"json",
                success:function(data){
                    $('#tbl1').empty();
                    for(i=0;i<data.length;i++){
                        var s = data[i];
                        $('#tbl1').append(
                            '<tr><td>' + s.code
                            + '</td><td>' + s.name
                            + '</td><td>' + s.price
                            + '</td></tr>');
                    }
                },
                error:function(){
                    //服务器出错之后的处理逻辑。
                }
            });
        }
    </script>
</head>
<body style="font-size:30px;font-style:italic;">
    <div id="d1">
```

```
<div id="d2">股票实时行情</div>
<div id="d3">
  <table width="100%" cellpadding="0"
    cellspacing="0">
    <thead>
      <tr><td>代码</td><td>名称</td><td>价格</td></tr>
    </thead>
    <tbody id="tb1">
    </tbody>
  </table>
</div>
</div>
</body>
</html>
```

Stock.java 文件代码如下：

```
package bean;

public class Stock {
  private String code;
  private String name;
  private double price;
  public String getCode() {
    return code;
  }
  public void setCode(String code) {
    this.code = code;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public double getPrice() {
    return price;
  }
  public void setPrice(double price) {
    this.price = price;
  }
}
```

ActionServlet.java 文件代码如下：

```
package web;

import java.io.IOException;
import java.io.PrintWriter;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import net.sf.json.JSONArray;
import bean.Stock;

public class ActionServlet extends HttpServlet {
```



```
public void service(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out = response.getWriter();
    String uri = request.getRequestURI();
    String action =
        uri.substring(uri.lastIndexOf("/"),uri.lastIndexOf("."));
    if(action.equals("/salary")){
        String flight = request.getParameter("eId");
        if(flight.equals("T10001")){
            out.println("实际工资：¥10000<br/>个税：¥2500");
        }else{
            out.println("实际工资：¥20000<br/>个税：¥5000");
        }
    }else if(action.equals("/quote")){
        Random r = new Random();
        List<Stock> stocks =
            new ArrayList<Stock>();
        for(int i=0;i<8;i++){
            Stock s = new Stock();
            s.setCode("60001" +r.nextInt(10));
            s.setName("中国国贸" + r.nextInt(10));
            DecimalFormat df = new DecimalFormat("0.00");
            double price = Double.parseDouble(
                df.format(100 * r.nextDouble()));
            s.setPrice(price);
            stocks.add(s);
        }
        JSONArray jsonArr =
            JSONArray.fromObject(stocks);
        String jsonStr = jsonArr.toString();
        System.out.println(jsonStr);
        out.println(jsonStr);
    }
    out.close();
}
}
```

课后作业

1. 实现如下图所示的页面效果，当鼠标移动到元素上时，样式发生改变，鼠标移出后，样式还原。

鼠标移动到元素上时，背景色改变。鼠标移出元素后，背景色还原。界面效果如图-1所示：

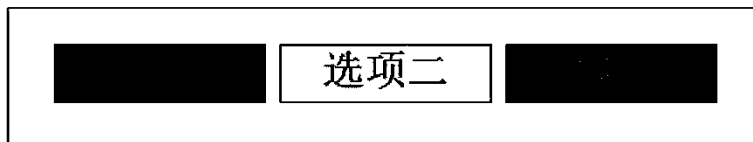


图 - 1

图-1 中选项二即为改变颜色后的效果。

2. 编写程序，实现类似于百度搜索时的自动完成功能。

使用 jQuery 实现类似于百度搜索时的自动完成功能，界面效果如图-2 所示：

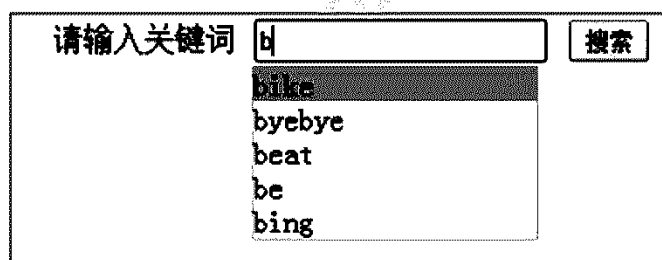


图 - 2