Jquery&Ajax

今日内容

> 1. 能够使用Jquery基本选择器
> 2. 能够使用Jquery对象完成dom操作
> 3. 掌握Jquery事件绑定
> 4. 理解原生JS异步通信原理
> 5. 掌握Jquery的异步通信
> 6. 能够完成异步通信案例

第1章 Jquery概述

1.1-jquery介绍

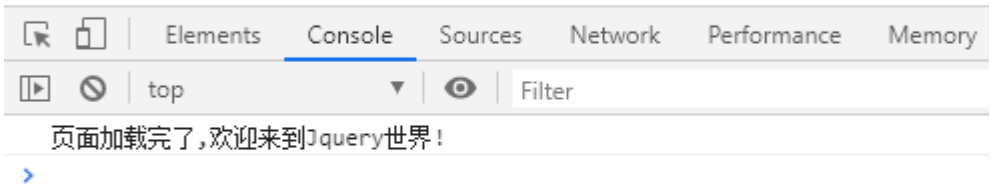
> jQuery是一个优秀的javascript的轻量级框架之一，封装了dom操作、事件、页面动画、异步操作等功能。
>
> 特别值得一提的是基于jquery的插件非常丰富，大多数前端业务场景都有其封装好的工具框架。

1.2-jquery版本

> - 1.x：兼容IE678，使用最为广泛的，官方只做BUG维护，功能不再新增。因此一般项目来说，使用1.x版本就可以了，最终版本：1.12.4 (2016年5月20日)
> - 2.x：不兼容IE678，很少有人使用，官方只做BUG维护，功能不再新增。如果不考虑兼容低版本的浏览器可以使用2.x，最终版本：2.2.4 (2016年5月20日)
> - 3.x：不兼容IE678，只支持最新的浏览器。除非特殊要求，一般不会使用3.x版本的，很多老的jQuery插件不支持这个版本。目前该版本是官方主要更新维护的版本
> - 开发版本与生产版本，命名为jQuery-x.x.x.js为开发版本，命名为jQuery-x.x.x.min.js为生产版本，开发版本源码格式良好，有代码缩进和代码注释，方便开发人员查看源码，但体积稍大。而生产版本没有代码缩进和注释，且去掉了换行和空行，不方便发人员查看源码，但体积很小。

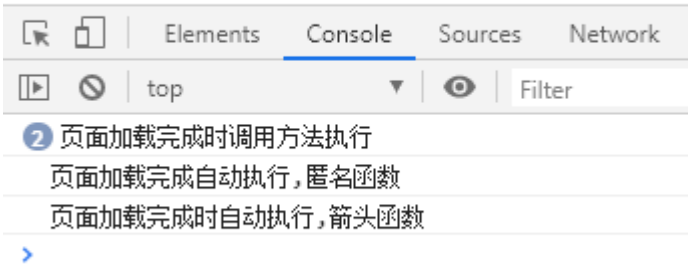
第2章Jquery基本语法

2.1-jquery环境引入




```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>jquery环境引入</title>
</head>
<body>

<!--引入jquery脚本库-->
<script src="../js/jquery-1.11.3.js"></script>

<script>
  $(document).ready(function(){
    console.log("页面加载完了,欢迎来到Jquery世界!");
  });//页面加载完成时
</script>

</body>
</html>
```

2.2-jquery基础语法

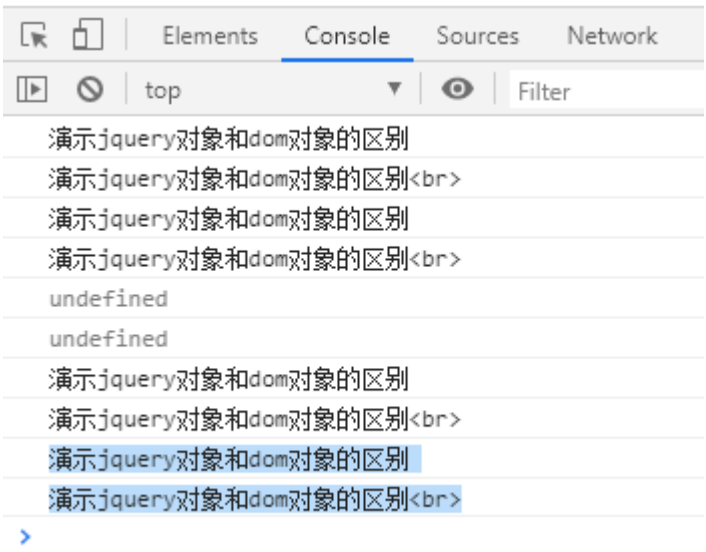



```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>jquery基础语法</title>
</head>
<body>
<!--
jQuery 语法是通过选取 HTML 元素，并对选取的元素执行某些操作。
    基础语法:  $(selector).action()
        $:使用美元符号来定义一个jquery对象
        selector: 匹配一个标签
        action: 执行jquery的方法
-->


<!--引入Jquery库文件-->
<script src="../js/jquery-1.11.3.js"></script>
<script>
  //页面加载完成时调用的方法
  let myf = function(){
    console.log("页面加载完成时调用方法执行");
  }

  //Jquery标准语法
  $(document).ready(myf);//页面加载完成时调用

  //Jquery简写
  $(myf);//同上， 页面加载完成时调用

  //简写(匿名函数) ,常用形式
  $(function(){
    console.log("页面加载完成自动执行,匿名函数");
  });


</script>

</body>
```

```
</html>
```

2.3-jquery对象与dom对象的区别

演示jquery对象和dom对象的区别

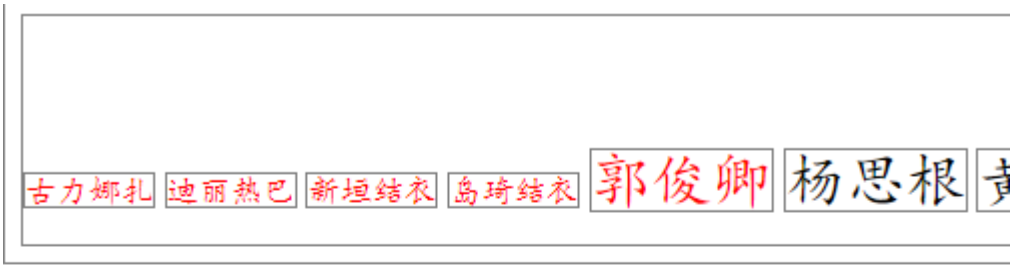


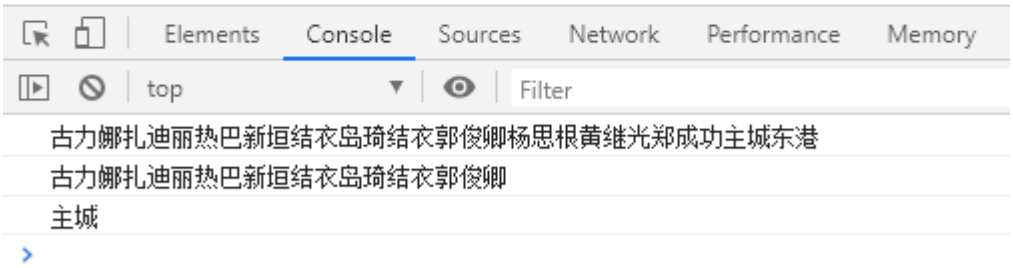

```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>jquery对象与dom对象</title>
  <!--引入Jquery库文件-->
  <script src="../js/jquery-1.11.3.js"></script>
</head>
<body>
<!--
jquery对象与dom对象定义
    dom对象:使用js形式获取的元素对象 例如 document.querySelector() 或
document.querySelectorAll()
    jquery对象: 使用 $()构造获得的元素对象  例如 $(selector)
```

```
使用注意:
  jquery对象只能使用jquery对象的属性和方法,不能使用dom对象的属性和方法
  dom对象只能使用dom对象的属性和方法,不能使用jquery对象的属性和方法

jquery对象与dom对象转换:
    dom对象 可以使用$构造 例如 $(dom对象)
    jquery对象 可以使用数组取索引值的形式 例如 $[index] 或 $.get(index)
-->

<div id="md">演示jquery对象和dom对象的区别<br/></div>

<script>
  //dom对象
  let div = document.querySelector("#md");//dom对象
  console.log(div.innerText);//文本
  console.log(div.innerHTML);//html
  //jquery对象
  let $div = $("#md");//jquery对象
  console.log($div.text());//文本
  console.log($div.html());//html

  //错误用法
  console.log($div.innerText);//undefined
  console.log($div.innerHTML);//undefined
  // console.log(div.text());//not a function
  // console.log(div.html());//not a function

  // dom对象可以使用$符构造成为jquery对象
  console.log($(div).text());//jquery对象
  console.log($(div).html());//jquery对象

  // jquery对象可以使用取索引的格式转为dom对象 [i]或get(i)
  console.log($div[0].innerText);//dom对象
  console.log($div[0].innerHTML);//dom对象


</script>

</body>
</html>
```

第3章 Jquery选择器

3.1-基本






```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>基本</title>
  <!--引入Jquery库文件-->
  <script src="../js/jquery-1.11.3.js"></script>
</head>
<body>
<!--
基本选择器:
    1.标签 根据标签匹配元素 格式 标签
    2.类 根据class的值匹配元素 class属性是给标签归类添加样式 格式 .class
    3.ID 根据id值匹配元素 id属性是标签的唯一标志 #id


-->


<span class="female" >古力娜扎</span>
<span class="female">迪丽热巴</span>
<span class="female">新垣结衣</span>
<span class="female">岛琦结衣</span>

<span class="female hero">郭俊卿</span>
<span class="male hero">杨思根</span>
<span class="male hero">黄继光</span>
```

```
<span class="hero">郑成功</span>

<span id="main">主城</span>
<span id="east">东港</span>
<script>
  //标签选择器
  let $span = $("span");//获取所有的span
  // console.log($span);//jquery对象
  console.log($span.text());//所有span的文本

  //类选择器
  let $female = $(".female");//jquery对象
  console.log($female.text());//所有class=female的span的文本

  //ID选择器
  let $main = $("#main");//jquery对象 id=main
  console.log($main.text());//文本

</script>
</body>
</html>
```

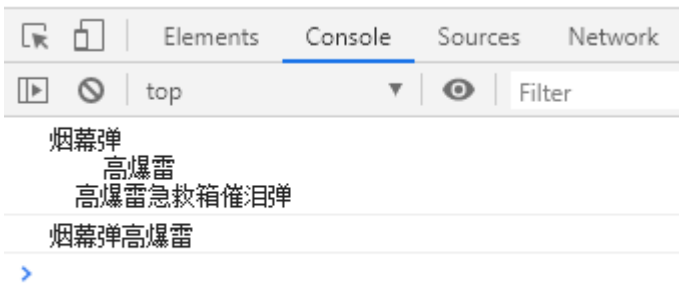
3.2-层级关系

烟幕弹

高爆雷

急救箱

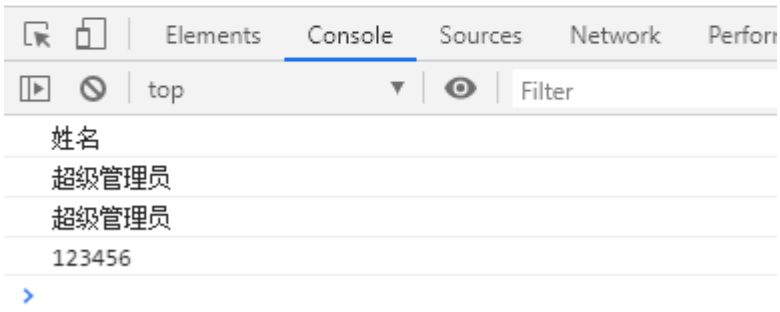
催泪弹




```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>层级关系</title>
```

```
  <!--引入Jquery库文件-->
  <script src="../js/jquery-1.11.3.js"></script>
</head>
<body>
<!--
关系(层级)
    E,F 并列 特点: 逗号
    E F 后代 特点: 空格
-->

<div>
  <span>烟幕弹</span>
  <p>
    <span id="gbl">高爆雷</span>
  </p>
</div>
<p class="jjx">急救箱</p>
<span id="cld">催泪弹</span>

<script >
  // E,F 并列 特点: 逗号
  console.log($("p,span").text());//获取所有的p,span的文本

  // E F 后代 特点: 空格
  console.log($("div span").text());//获取div的后代span的文本



</script>


</body>
</html>
```

3.3-属性

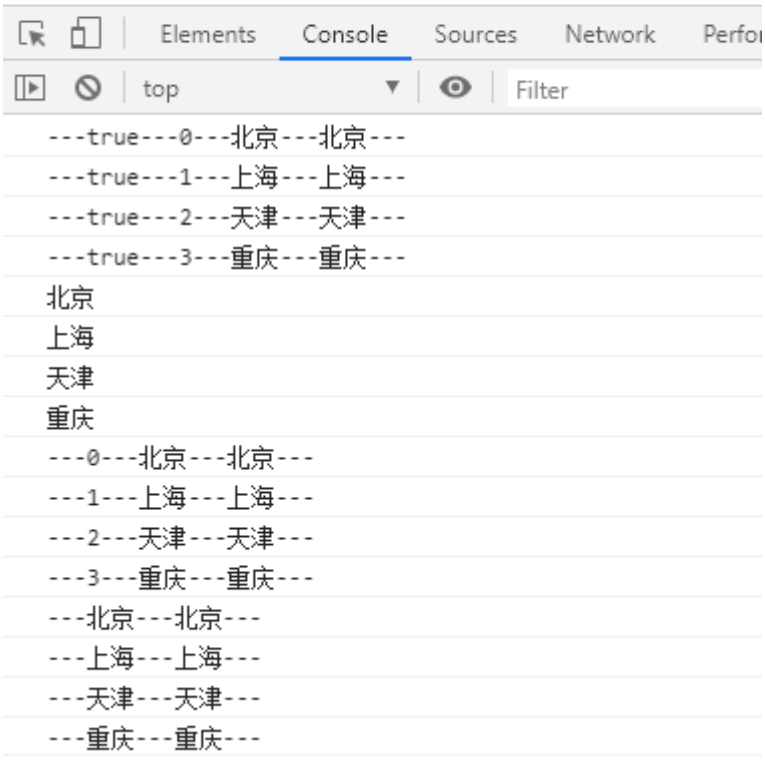



```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>属性</title>
  <!--引入Jquery库文件-->
  <script src="../js/jquery-1.11.3.js"></script>
</head>
<body>
<!--
属性选择器
    E[attribute=value]
-->

<label for="userName">姓名</label>
<input id="userName" type="text" name="userName" value="超级管理员"/><br/>
<label>密码</label>
<input type="password" name="userPass" value="123456"/><br/>

<script>

  //获取有for属性的label标签
  console.log($("label[for]").text());

  //获取type='text'的input标签
  console.log($('input[type="text"]').val());// .val() jquery取value值
  console.log($('input[type="text"]')[0].value);//同上 .value dom取value值

  //获取type="password"且name="userPass"的input标签
  console.log($("input[type='password'][name='userPass']").val());//jquery取value值
```

```
</script>

</body>
</html>
```

3.4-对象遍历

- 北京
- 上海
- 天津
- 重庆




```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>对象遍历</title>
  <!--引入Jquery库文件-->
  <script src="../js/jquery-1.11.3.js"></script>
</head>
<body>
<!--
```

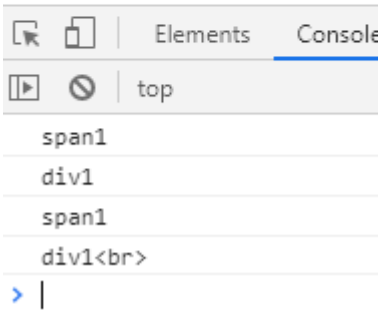
```
jquery对象的遍历
    1.$.each() 用法示例 $.each(function(i,e){}) 或 $.each((i,e)=>{}) 注意 e是dom对象
    2.fori $.length 表示jquery对象中元素的个数 $[index]
    3.forof $.toArray()把jquery对象转为dom对象数组
-->
<ul id="city">
  <li>北京</li>
  <li>上海</li>
  <li>天津</li>
  <li>重庆</li>
</ul>
<script>
  let $lis = $("li");//获取所有li
  // 1.$.each() 匿名函数
  $lis.each(function(i,e){
    console.log(`---${this === e}---${i}---${e.innerText}---${$(this).text()}---`);
  });
  //重点掌握这一种
  $lis.each(function(){
    console.log($(this).text());//每一个jquery对象元素
  });
  console.log(this);//window
  // 1.$.each() 箭头函数
  $lis.each((i,e)=>{
    console.log(`---${this === window}---${i}---${e.innerText}---${$(e).text()}---`);
  });
  // 2.fori
  for(let i =0;i<$lis.length;i++){
    console.log(`---${i}---${$lis[i].innerText}---${$($lis[i]).text()}---`);

  }
  // 3.forof
  for(let e of $lis.toArray()){//$.toArray()把jquery对象转为dom对象数组
    console.log(`---${e.innerText}---${$(e).text()}---`);
  }

</script>
</body>
</html>
```

第4章 Jquery的dom操作

4.1-jquery操作内容

小宝剑

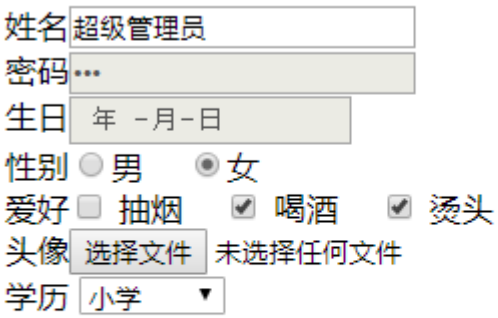
小宝剑

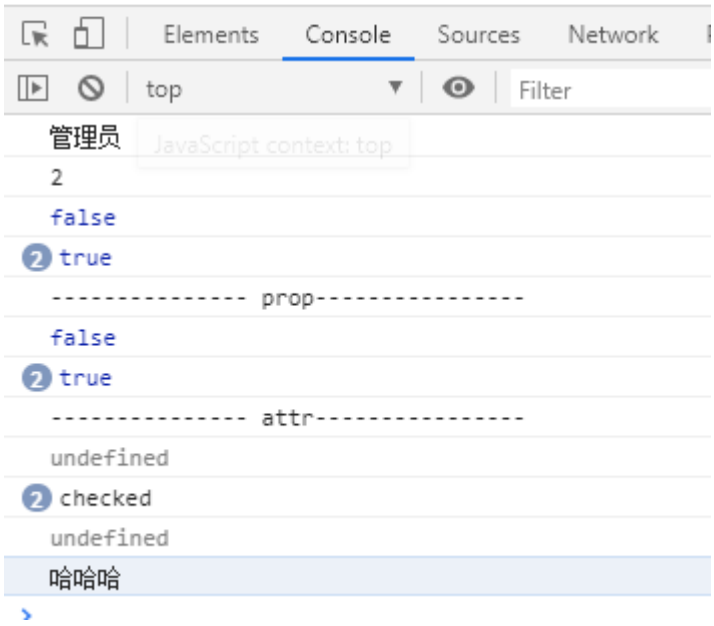



```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>jquery操作内容</title>
  <!--引入Jquery库文件-->
  <script src="../js/jquery-1.11.3.js"></script>
</head>
<body>
<!--
jquery操作内容
    1. text() 获取或修改文本内容 类似于 dom.innerText
    2. html() 获取或修改html内容 类似 dom.innerHTML
查询jquery手册完成
    1.获取纯文本内容
    2.获取html内容
    3.设置纯文本内容
    4.设置html内容

-->

<span>span1</span>
<div>div1<br/></div>

<script>
  let $spans = $("span");//获取所有span
  let $divs = $("div");//获取所有div

  // 1.获取纯文本内容
  console.log($spans.text());//文本
  console.log($divs.text());//文本
  // 2.获取html内容
  console.log($spans.html());//html
  console.log($divs.html());//html
```

```
  // 3.设置纯文本内容
  $spans.text("<h2>小宝剑</h2>");//设置文本内容
  $divs.text("<h2>小宝剑</h2>");//设置文本内容
  // 4.设置html内容
  $spans.html("<h2>小宝剑</h2>");//设置html内容
  $divs.html("<h2>小宝剑</h2>");//设置html内容
</script>

</body>
</html>
```

4.2-jquery操作属性



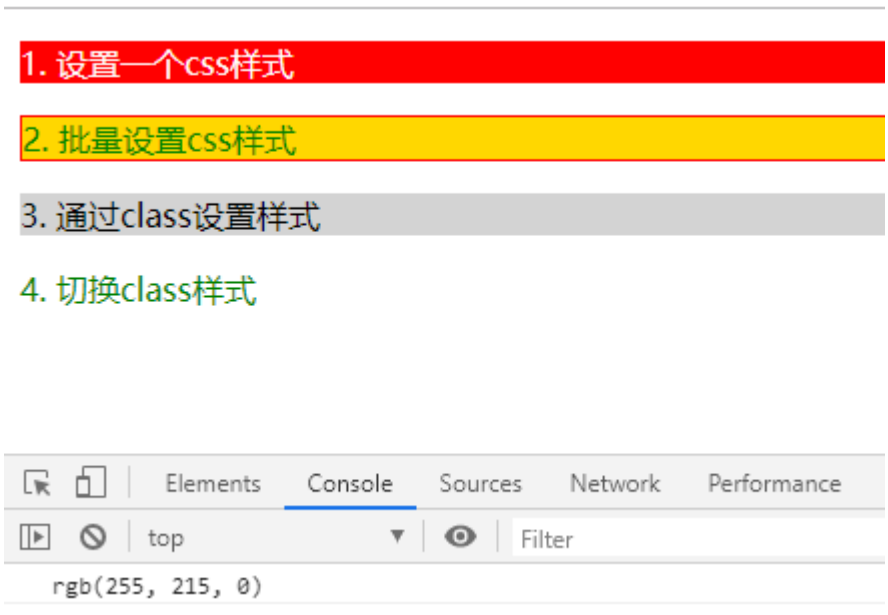



```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>jquery操作属性</title>
  <!--引入Jquery库文件-->
```

```
  <script src="../js/jquery-1.11.3.js"></script>
</head>
<body>
<!--
jquery操作属性
    1. val() 获取或修改表单控件的value值 等于dom.value
    2. prop() 获取或修改原生属性值 等于 dom.properties
    3. attr() 获取或修改自定义属性值 等于 dom.setAttribute() dom.getAttribute()



查询jquery手册完成
    1.获取表单控件的value值(所有文本和下拉列表)
    2.获取单选复选框的选中状态(原生属性)
    3.获取自定义属性值
总结:
    val() 操作表单控件的value值
    prop() 操作原生属性
    attr() 操作自定义属性
-->
<form action="#" method="get">
  <input type="text" name="userId"  value="007" hidden="hidden" id="userId"/><br/>
  姓名<input type="text" name="userName"  value="管理员" id="userName" data-msg="哈哈哈"/>
<br/>
  密码<input type="password" name="userPass" value="123" disabled id="userPass"/><br/>
  生日<input type="date" name="userBirth" disabled="disabled" id="userBirth"/><br/>
  性别<input type="radio" name="gender" value="male" id="male"/>男 
  <input type="radio" name="gender" value="female" checked="checked" id="female"/>女<br/>
  爱好<input type="checkbox" name="hobby" value="smoke" id="smoke"/>
  <label for="smoke">抽烟 </label>
  <input type="checkbox" name="hobby" value="drink" checked id="drink"/>
  <label for="drink">喝酒 </label>
  <input type="checkbox" name="hobby" value="perm" checked="checked" id="perm"/>
  <label for="perm">烫头 </label><br/>
  头像<input type="file" name="userPic"/><br/>
  学历
  <select name="userEdu" id="userEdu">
    <option value="">-请选择-</option>
    <option value="1">小学</option>
    <option value="2" selected>初中</option>
    <option value="3">高中</option>
  </select>
</form>
<script>
  // 1.获取表单控件的value值
  let $un = $("#userName");//姓名文本框
  //获取文本框的value值
  console.log($un.val());
  //设置文本框的value值
  $un.val("超级管理员");
  //获取下拉列表的value值
  console.log($("#userEdu").val());
  //设置下拉列表的value值
```

```
  $("#userEdu").val(1);//选中小学
  // 2.获取单选复选框的选中状态(原生属性)
  //js方式
  let $hobbys = $('input[name="hobby"]');//所有复选框的jquery对象
  $hobbys.each(function(){
    console.log(this.checked);// 选中 true 未选中 false
  });
  console.log(`--------------- prop----------------`);
  //原生属性
  $hobbys.each(function(){
    // console.log(this.checked);// 选中 true 未选中 false
    console.log($(this).prop("checked"));//同上 选中 true 未选中 false
  });
  console.log(`--------------- attr----------------`);
  //自定义属性
  $hobbys.each(function(){
    console.log($(this).attr("checked"));// 不要这么使用
  });

  // 3.获取自定义属性值
  console.log($un.prop("data-msg"));;//undefined
  console.log($un.attr("data-msg"));;//获取自定义属性值

</script>
</body>
</html>
```

4.3-jquery操作样式

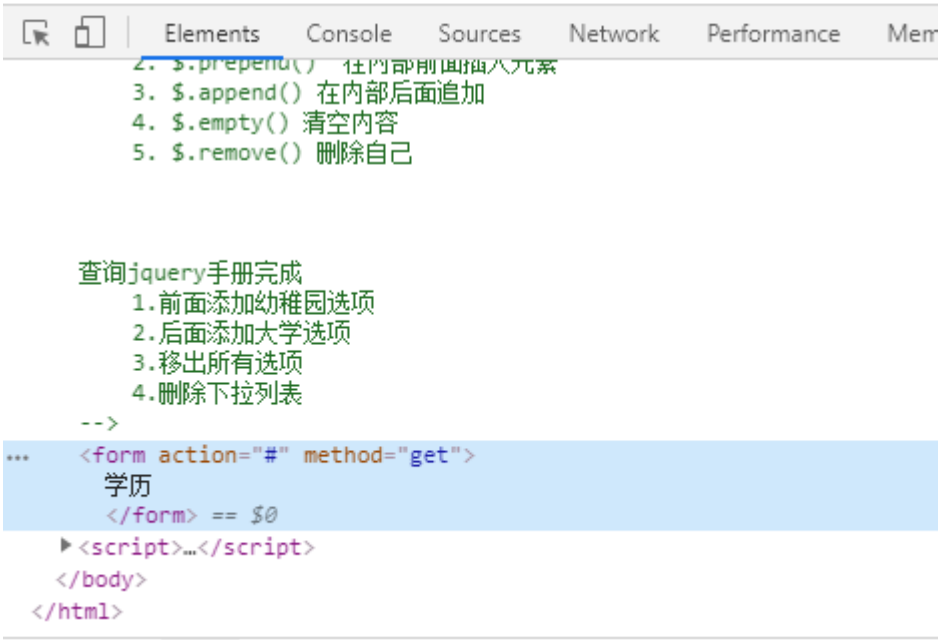



```
<!DOCTYPE html>
<html lang="zh">
```

```
<head>
  <meta charset="UTF-8">
  <title>jquery操作样式</title>
  <!--引入Jquery库文件-->
  <script src="../js/jquery-1.11.3.js"></script>
</head>
<body>
<!--
jquery操作样式
    1. css() 获取或者修改CSS样式 用法
            css(样式名) 获取一个样式 等于 dom.style.驼峰样式名
            css(样式名,样式值) 设置一个样式 dom.style.驼峰样式名=样式值
            css({样式名:样式值,样式名:样式值}) 批量设置样式 dom.style.cssText = 样式名:样式值;样式
名:样式值;
    2. addClass() 添加一个样式 等于 dom.classList.add()
    3. removeClass() 移除一个样式 dom.classList.remove()
    4. toggleClass() 切换一个样式dom.classList.toggle()
-->
<style>
  #p1{ background-color: red;}
  .mp{color:green}
  .mpp{background-color: lightgray;}
</style>
<p id="p1">1. 设置一个css样式</p>
<p id="p2" >2. 批量设置css样式</p>
<p id="p3" class="mpp">3. 通过class设置样式</p>
<p id="p4">4. 切换class样式</p>
<script>
  let $p1 = $('#p1');//获取p1
  let $p2 = $('#p2');//获取p2
  let $p3 = $('#p3');//获取p3
  let $p4 = $('#p4');//获取p4
  // 1. css() 获取或设置css样式
  //设置一个样式
$p1.css("color","white");//设置字体白色
  //批量设置样式
  $p2.css({"color":"green","background-color":"gold","border":"1px solid red"});//批量设置
  //获取一个样式
  console.log($p2.css("background-color"));
  // 2. addClass() 添加一个class样式
  $p4.addClass("mp");//添加一个样式
  $p4.addClass("mpp");//添加一个样式
  // 3. removeClass() 移除一个class
  $p4.removeClass("mpp");
  // 4. toggleClass() 切换一个class
  $p4.toggleClass("mpp");//切换 无则添加
  $p4.toggleClass("mpp");//切换 有则删除
</script>
</body>
</html>
```

4.4-jquery操作元素

学历




```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>jquery操作元素</title>
  <!--引入Jquery库文件-->
  <script src="../js/jquery-1.11.3.js"></script>
</head>
<body>
<!--
jquery操作元素
    1. $(标签) 创建一个标签
    2. $.prepend()  在内部前面插入元素
    3. $.append() 在内部后面追加
    4. $.empty() 清空内容
    5. $.remove() 删除自己
```

```
查询jquery手册完成
    1.前面添加幼稚园选项
    2.后面添加大学选项
    3.移出所有选项
    4.删除下拉列表
-->
<form action="#" method="get">
  学历
  <select name="userEdu" id="userEdu">
    <option value="1">小学</option>
    <option value="2">初中</option>
    <option value="3">高中</option>
  </select>
</form>
<script>
  let $select = $("#userEdu");//下拉列表对象
  // 1.前面添加幼稚园选项
  let $option = $('<option value="0">幼稚园</option>');
  $select.prepend($option);//在内部前面插入
  // 2.后面添加大学选项
  $select.append('<option value="4">大学园</option>');//在内部后面追加
  // 3.移出所有选项
  $select.empty();//清空内容
  // 4.删除下拉列表
  $select.remove();//删除自己
</script>
</body>
</html>
```

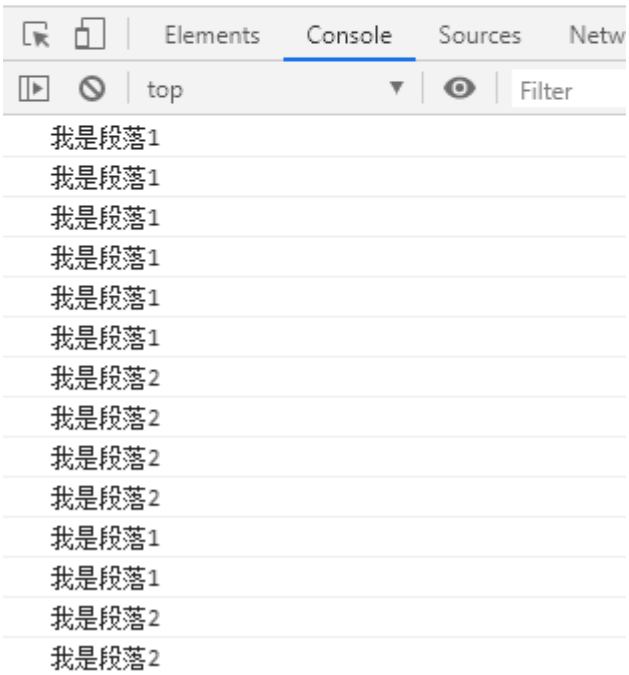
第5章 Jquery 绑定事件

5.1-jquery绑定事件

我是段落1

我是段落2

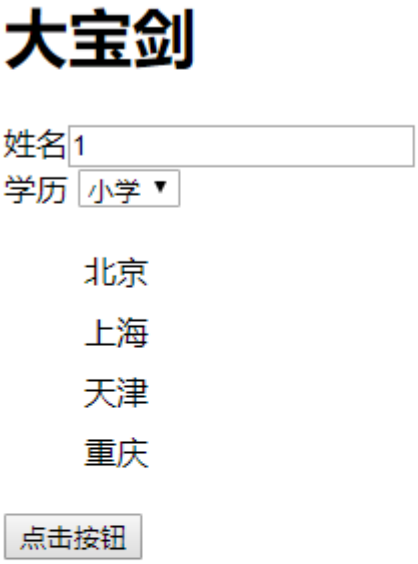
我是段落3解绑定事件

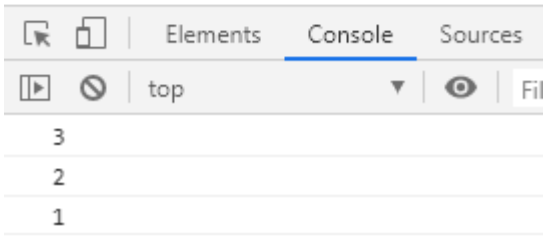



```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>绑定事件</title>
  <!--引入Jquery库文件-->
  <script src="../js/jquery-1.11.3.js"></script>
</head>
<body>
<!--
jquery绑定事件
    1. $.event(函数) 把event事件与函数绑定    注意: $.event() 手动触发一次事件
    2. $.on("event",函数) 把event事件与函数绑定
    3. $.off("event") 解除event事件绑定

jquery事件与dom事件写法区别
    jquery 都不要加on 例如 $.click(函数)
    dom  都需要加on 例如 dom.onclick = 函数
-->
<p id="p1">我是段落1</p>
<p id="p2">我是段落2</p>
```

```
<p id="p3" onclick="offevent()">我是段落3解绑定事件</p>
<script>
  //1. $.event(函数) 点击p1时获取其纯文本内容
  $("#p1").click(function(){
    console.log(this.innerText);//p1的文本
    console.log($(this).text());//同上 p1的文本
  });
  //手动触发一次p1的点击事件
  $("#p1").click();//手动触发一次单击事件
  //2. $.on("event",函数);  点击p2时获取其纯文本内容
  $("#p2").on("click",function(){
    console.log(this.innerText);//文本内容
    console.log($(this).text());//文本内容
  });

  //3. $.off("event")解绑定事件 解除p1, p2 的单击事件绑定
  function offevent(){
    $("#p1").off("click");//解除p1单击事件绑定
    $("#p2").off("click");//解除p2单击事件绑定
  }

</script>
</body>
</html>
```

5.2-常用事件






```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>常用事件</title>
  <!--引入Jquery库文件-->
  <script src="../js/jquery-1.11.3.js"></script>
  <style>
    #city{
      list-style: none;/*列表项目图标类型:不显示图标*/
      line-height: 30px;/*字体行高30px*/
    }
  </style>
</head>
<body>
<!--
jquery常用事件
    jquery事件基本与dom事件一样,并且在dom事件基础上做了优化

查询jquery手册完成
    1. ready 页面加载完成
    2. blur 失去焦点
    3. change 表单控件的value值改变时
    2. mouseover mouseout hover 鼠标移入移出
    5. click 单击
```

```
注意:前面没有dom事件的on
-->
<h1 id="h1">我是大标题</h1>
姓名<input type="text" name="userName"  id="userName" /><br/>
学历
<select name="userEdu" id="userEdu">
  <option value="1">小学</option>
  <option value="2">初中</option>
  <option value="3">高中</option>
</select>
<ul id="city">
  <li>北京</li>
  <li>上海</li>
  <li>天津</li>
  <li>重庆</li>
</ul>
<button id="btn">点击按钮</button>
<script>
  // 1. ready 页面加载完成,修改标题内容
  $(function(){//页面加载完成时
    $("#h1").text("大宝剑");//修改文本内容
  });
  // 2. blur 文本框失去焦点时获取其value值
  $("#userName").blur(function(){
    console.log($(this).val());//文本框的value值
  });

  // 3. change 表单控件value值改变时, 获取下拉列表选中的值
  $("#userEdu").change(function(){
    console.log($(this).val());//下拉列表的value值
  });
  // 2. mouseover mouseout hover 无序列表 鼠标移入移出改变背景颜色
  //移入移出分开写法
  // $("#city li").mouseover(function(){//鼠标移入事件
  //   $(this).css("background-color","pink");//背景粉色
  // });
  // $("#city li").mouseout(function(){//鼠标移出事件
  //   $(this).css("background-color","white");//背景白色
  // });
  //移入移出二合一写法
  $("#city li").hover(
          function(){//鼠标移入事件
            $(this).css("background-color","pink");//背景粉色
          }
          ,
          function(){//鼠标移出事件
            $(this).css("background-color","white");//背景白色
          }
  );
  // 5. click 鼠标单击按钮时 把文本框的值赋值给下拉列表
  $("#btn").click(function(){//鼠标点击按钮事件
    $("#userEdu").val($("#userName").val());//把文本的值赋值给下拉列表
```

```
  });

</script>
</body>
</html>
```

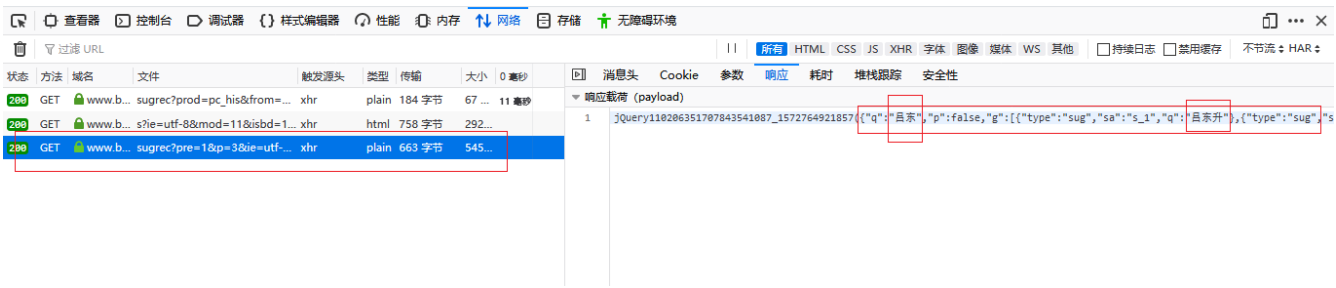
第6章 Ajax概述

6.1-作用介绍

> Ajax可以请求并获取服务器的数据,来完成与服务的通信.

6.2-适用场景

> 百度搜索

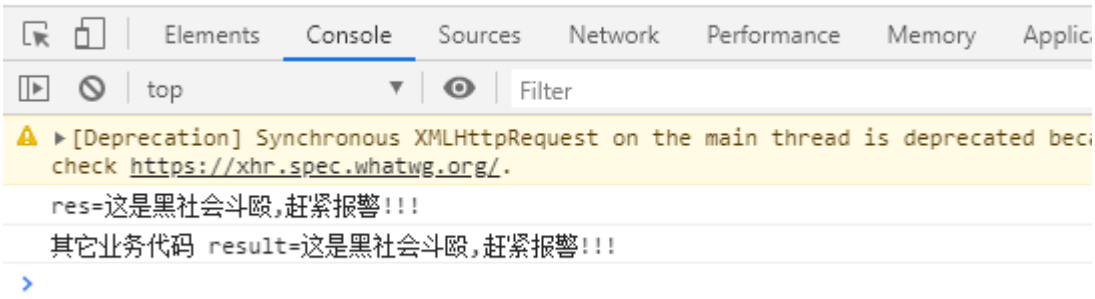


> 注册账号验重

第7章 原生JS异步通信

7.1- js原生ajax实现

这一群人正在军训




```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>js原生ajax实现</title>
```

```
</head>
<body>
<!--
js原生ajax实现原理(理解)
XMLHttpRequest 一个可以向服务器发送数据,并能接受返回数据的js对象. 常用方法和属性
  open(method, url, boolean async) : 创建打开一个连接 参数method请求方式 url服务器地址 async是否异步
  send(): 开始发送数据
  readyState: 请求就绪状态 值为4时说明请求准备完成
  onreadystatechange: 监听请求就绪状态的改变
  status: 请求完成情况 值为200时说明请求成功完成
  responseText:服务器返回的文本


查询手册完成功能:
    点击图片时,请求服务器获取数据,把接收的数据显示在页面
-->
<img src="../img/timg.gif" onclick="doAjax()">
<p id="describe">这一群人正在军训</p>
<script type="text/javascript">
  let http  = new XMLHttpRequest();//一个可以向服务器发送数据并接收服务器返回数据的对象

  let result = null;//服务器返回的数据

  http.onreadystatechange = function(){//请求就绪状态改变时
    // console.log(http.readyState);//请求就绪状态
    if(http.readyState == 4 && http.status == 200){//请求成为完成
      let res = http.responseText;//服务器返回的文本数据
      result = res;//服务器返回的数据
      console.log("res="+res);
    }

  }

  //点击图片时,请求服务器获取数据,把接收的数据显示在页面
  function doAjax(){
    //发送ajax请求
    http.open("get","../data/info.txt",true);//创建打开一个异步请求
    // http.open("get","../data/info.txt",false);//创建打开一个同步请求
    http.send();//开始发送数据

    //其它业务代码
    console.log("其它业务代码 result="+result);//发送ajax之后的其它js业务代码
    //同步 发送ajax请求之后其它js业务代码必须等待ajax请求成功以后才能执行
    //异步 发送ajax请求之后的其它js业务代码的执行,与ajax请求的执行,没有任何关系.
    //总结: ajax异步请求时服务器返回的数据只能在ajax请求完成的回调函数中使用.

  }


</script>
</body>
</html>
```

第8章 Jquery的Ajax插件

8.1-jquery的Ajax语法

```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>jquery的Ajax语法</title>
</head>
<body>
<!--
jquery ajax语法:
    $.ajax(配置);
    配置是个JSON对象 常用属性有
      type 请求方式 get post
      url 服务器地址
      data 发送给服务器的数据
      dataType 预期服务器将要返回的数据类型 text json
      async 是否异步 默认true异步
      success 请求成功时的回调函数
      error 请求失败时的回调函数

使用示例:

//发送ajax请求

$.ajax({
type:"get",
url:"../data/student.txt",
data:{name:"张三",pass:123},
dataType:"text",
async:true,
success:function(response){},
error:function(error){}
});


简写形式(了解):
语法: $.get(url,data,success); 使用示例:
$.get("test.php", { name: "张三", pass: "123" },
  function(response){
    console.log(response);
  });
语法: $.post(url,data,success); 使用示例:
$.post("test.php", { name: "张三", pass: "123" },
   function(response){
     console.log(response);
   });
```

```
 -->
</body>
</html>
```

8.2-jquery的ajax示例

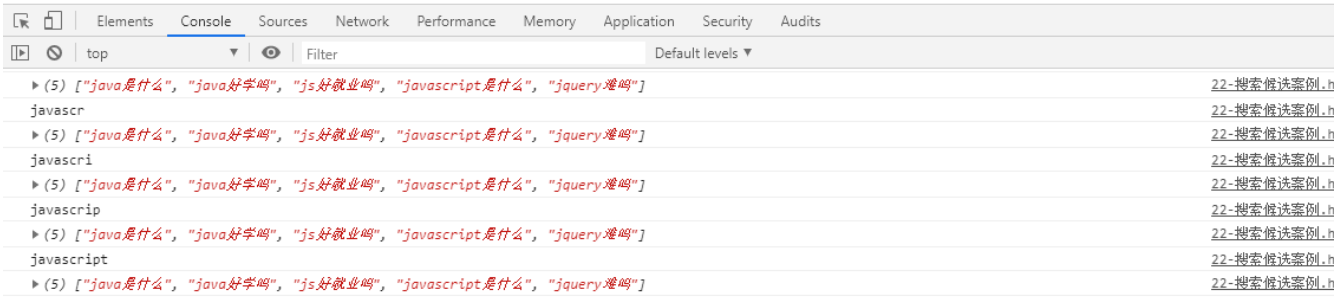

```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>jQuery的ajax示例</title>
  <!--引入Jquery库文件-->
  <script src="../js/jquery-1.11.3.js"></script>
</head>
<body>
<!--
jQuery的ajax示例
    1. 点击图片请求服务器(用本地json文件模拟服务器),并把接受到的数据显示在页面
 -->

<img src="../img/timg.gif" onclick="doAjax()">
```

```
<p id="describe">这一群人正在军训</p>

<script type="text/javascript">
  //1. 点击图片请求服务器(用本地文件模拟服务器),并把接受到的数据显示在页面
  function doAjax(){


    let result = null;//服务器返回的数据

    //发送ajax请求
    $.ajax({
      type:"get",//本地文件只能使用get方式
      url:"../data/student.txt",//服务器地址
      // dataType:"json",//(了解)预期服务器返回json,走一次JSON.parse()
      success:function(response){//成功回调
        console.log(response);
        let resObj = JSON.parse(response);//把服务器返回的字符串转为对象
        console.log("resObj.desc="+resObj.desc);//描述信息
        result =resObj.desc;//把服务器返回的字符串
        $("#describe").text(resObj.desc);//把服务器返回的数据显示在页面
      },
      error:function(error){//失败回调
        console.log(error);
      }
    });

    //其它业务代码
    console.log("其它业务代码 result="+result);//发送ajax之后的其它js业务代码
    //同步 发送ajax请求之后其它js业务代码必须等待ajax请求成功以后才能执行
    //异步 发送ajax请求之后的其它js业务代码的执行,与ajax请求的执行,没有任何关系.
    //总结: ajax异步请求时服务器返回的数据只能在ajax请求完成的回调函数中使用.



  }


</script>
</body>
</html>
```

第9章 异步通信综合案例

> 使用本地文件模拟服务器
>
> 实现百度的搜索候选功能

9.1-案例-搜索候选




```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>案例-搜索候选功能</title>
  <!--引入Jquery库文件-->
  <script src="../js/jquery-1.11.3.js"></script>
  <style type="text/css">
    .content {
      width: 643px;
      margin: 200px auto;
      text-align: center;
    }

    input[type='text'] {
      width: 534px;
      height: 40px;
      font-size: 14px;
    }

    input[type='button'] {
      width: 100px;
      height: 46px;
      background: #38f;
      border: 0;
      color: #fff;
      font-size: 15px
    }

    .show {
      position: absolute;
```

```
      width: 516px;
      border: 1px solid #efefef;
      border-top: 0;
      display: none;
      text-align: left;
      color: darkgray;
      line-height: 20px;
      padding: 5px 10px;
    }
  </style>
  <script type="text/javascript">
    //当键盘按键弹起时调用
    function searchWord(obj) {
      console.log(obj.value);//输入的字符
      $("#show").empty();//清空内容
      if(obj.value.length ==0)return;//输入框中没有字符

      let reg = new RegExp("^"+obj.value);//匹配以输入字符开头的字符串

      //发送ajax请求
      $.ajax({
        type:"get",
        url:"../data/data.txt",
        data:{msg:obj.value},
        success:function(res){//成功回调
          // console.log(res);//先看看是string类型还是object类型
          let arr = JSON.parse(res);//把服务器返回的字符串转为object
          console.log(arr);
          //迭代数组arr,把每个元素追加到div中
          for(let s of arr){
            if(reg.test(s)){//以输入字符开头的候选项
              $("#show").append("<div>"+s+"</div>");//向div中追加内容
            }
          }
          // $("#show").css("display","block")//显示div
          $("#show").show();//显示 同上
          // $("#show").hide();//隐藏
        },
        error:function(err){//失败回调
          console.log(err);
        }
      });

    }
  </script>
</head>
<body>
<div class="content">

  <input type="text" name="word" onkeyup="searchWord(this)">
  <input type="button" value="搜索一下">
  <div class="show" id="show"></div>
</div>
```

```
</body>
</html>
```