优极限

“极限教育，挑战极限”

www.yjxxt.com

极限教育，挑战极限。优极限是一个让 95%的学生年薪过 18 万的岗前培训公司，让我们的学员具备优秀的互联网技术和职业素养，勇攀高薪，挑战极限。公司位于上海浦东，拥有两大校区，共万余平。累计培训学员超 3 万名。我们的训练营就业平均月薪 19000，最高年薪 50 万。

核心理念：让学员学会学习，拥有解决问题的能力，拿到高薪职场的钥匙。

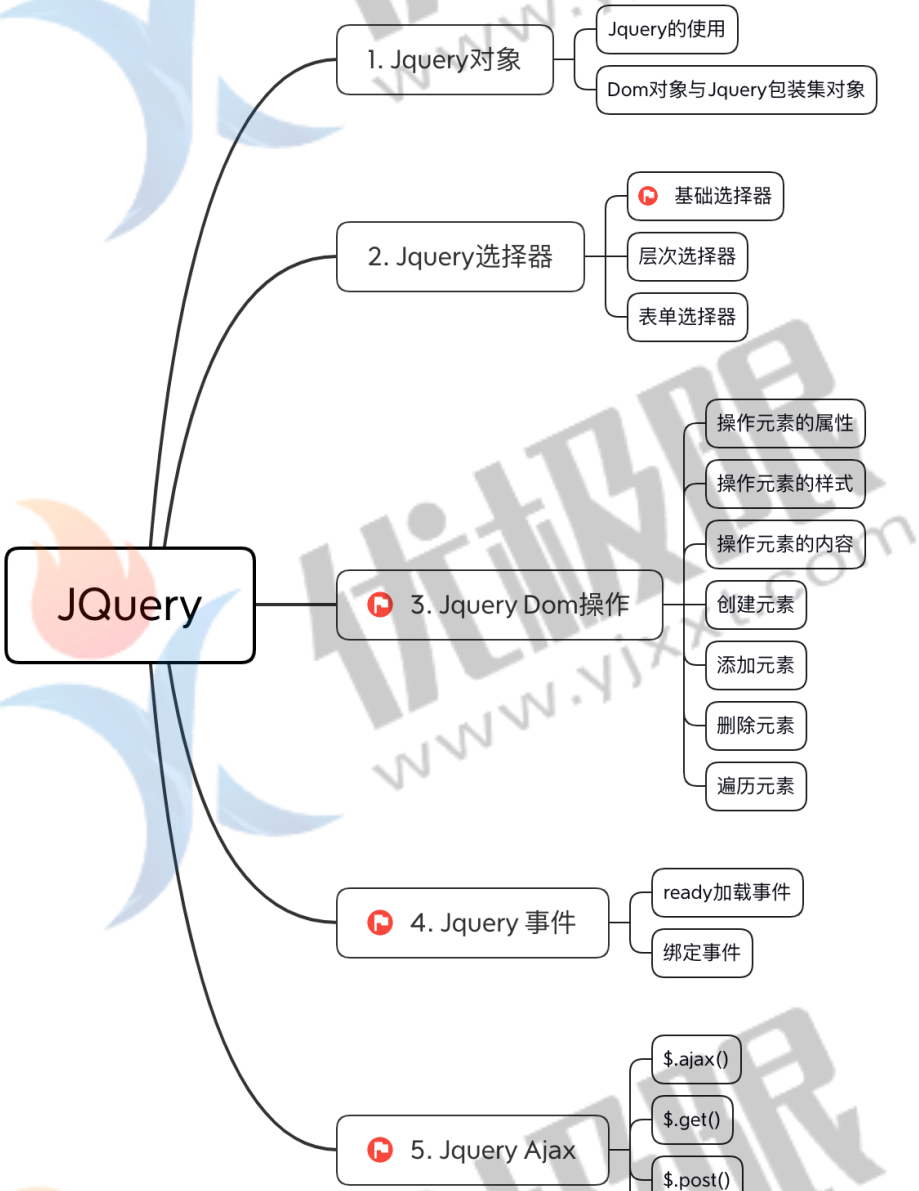
项目驱动式团队协作、一对一服务、前瞻性思维、教练式培养模型-培养你成为就业明星。首创的老学员项目联盟给学员充分的项目、技术支撑，利用优极限平台这根杠杆，不断挑战极限，勇攀高薪，开挂人生。

扫码关注优极限微信公众号：

（获取最新技术相关资讯及更多源码笔记）

JQuery

1. 主要内容



2. Jquery对象

jQuery 是一套兼容多浏览器的 javascript 脚本库.。核心理念是写得更少，做得更多，使用 jQuery 将极大的提高编写 javascript 代码的效率，帮助开发者节省了大量的工作，让写出来的代码更加优雅，更加健壮，"如虎添翼"。同时网络上丰富的 jQuery 插件，也让我们的工作变成了"有了 jQuery，一切 so easy。" -- 因为我们已经站在巨人的肩膀上了。

jQuery 在 2006 年 1 月由美国人 John Resig 在纽约的 barcamp 发布，吸引了来自世界各地的众多 JavaScript 高手加入，由 Dave Methvin 率领团队进行开发。如今，jQuery已经成为最流行的 javascript 框架，在世界前 10000 个访问最多的网站中，有超过 55%在使用 jQuery。

2.1. Jquery的下载与安装

2.1.1. 下载

http://jquery.com/ 下载

2.1.2. 版本

jQuery 2.x has the same API as jQuery 1.x, but does not support Internet Explorer 6, 7,or 8. (不支持 ie6 7 8,如果需要下载 1.X)

（1）完整版 : jquery-2.1.4.js -->学习版本(学习源码 向高手学习是最好学习方法)

（2）压缩版 : jquery-2.1.4.min.js -->开发版本(压缩版，减少传输)

目前使用版本：jquery-3.4.1.js

2.1.3. 优点

（1）提供了强大的功能函数

（2）解决浏览器兼容性问题

（3）实现丰富的 UI 和插件

（4）纠正错误的脚本知识

.....

2.1.4. 安装

在页面引入即可

```
<script src="js/jquery-3.4.1.js" type="text/javascript" ></script>
```

2.2. Jquery核心

$ 符号在 jQuery 中代表对 jQuery 对象的引用， "jQuery"是核心对象。通过该对象可以获取jQuery对象，调用jQuery提供的方法等。只有jQuery对象才能调用jQuery提供的方法。

```
$ <==> jQuery
```

2.3. Dom对象 与 Jquery包装集对象

明确 Dom 对象和 jQuery 包装集的概念， 将极大的加快我们的学习速度。原始的 Dom 对象只有 DOM 接口提供的方法和属性，通过js代码获取的对象都是 Dom 对象；而通过 jQuery 获取的对象是 jQuery 包装集对象，简称jQuery对象，只有jQuery对象才能使用jQuery提供的方法。

2.3.1. Dom对象

javascript 中获取 Dom 对象， Dom 对象只有有限的属性和方法：

```
var div = document.getElementById("testDiv");
var divs = document.getElementsByTagName("div");
```

2.3.2. Jquery包装集对象

可以说是 Dom 对象的扩充。在 jQuery 的世界中将所有的对象， 无论是一个还是一组，都封装成一个 jQuery 包装集，比如获取包含一个元素的 jQuery 包装集：

```
var jQueryObject = $("#testDiv");
```

2.3.3. Dom对象 转 Jquery对象

Dom 对象转为 jQuery 对象，只需要利用 `$()` 方法进行包装即可

```
var domDiv = document.getElementById('mydiv');  // 获取Dom对象
mydiv = $(domDiv);
```

2.3.4. Jquery对象 转 Dom对象

jQuery 对象转 Dom 对象，只需要取数组中的元素即可

```
// 第一种方式 获取jQuery对象
var jqueryDiv = jQuery('#mydiv');
// 第二种方式 获取jQuery对象
jqueryDiv = $('#mydiv');
var dom = jqueryDiv[0]; // 将以获取的jquery对象转为dom
```

通过遍历 jQuery 对象数组得到的对象是 Dom 对象，可以通过 `$()` 转为 jQuery 对象

```
$('#mydiv').each(function() {//遍历
     var jquery = $(this);
});
```

案例：

```
<div id="mydiv">write less, do more</div>

<script type="text/javascript">
  console.log("-------------获取dom对象-----------------")
  // dom对象
  var domDiv = document.getElementById("mydiv");
  console.log(domDiv);

  console.log("-------------获取jquery对象-----------------")
  // 获取jquery对象
  // 第一种方式
  var jqueryDiv = jQuery('#mydiv');
  console.log(jqueryDiv);
  // 第二种方式
  jqueryDiv = $('#mydiv');
  console.log(jqueryDiv);

  console.log("-------------dom转jquery-----------------")
  // dom转jquery包装集/对象
  var obj = $(domDiv);
  console.log(obj);

  console.log("-------------jquery转dom-----------------")
  // jquery对象转dom对象
  var dom = $('#mydiv')[0]; // 获取jquery对象转为dom
  // 或
  var dom2 = jqueryDiv[0]; // 将jquery对象转为dom
  console.log(dom);
  console.log(dom2);

  /* this代表了dom对象，不是jquery对象 */
  console.log("-------------dom转jquery-----------------")
  $('#mydiv').each(function() {
   // 通过id选择器选择了id为mydiv的所有元素然后进行遍历
   // 那么遍历出的每个元素就是id为mydiv的标签元素
   // 而this就代表了当前的这个元素
    var jquery = $(this);
  });

  console.log("-------------jquery转dom-----------------")
  $('#mydiv').each(function() {
    var dom3 = this;
  });
</script>
```

3. Jquery选择器

和使用 JS 操作Dom一样，获取文档中的节点对象是很频繁的一个操作，在jQuery中提供了简便的方式供我们查找|定位元素，称为jQuery选择器，选择器可以说是最考验一个人 jQuery 功力的地方，通俗的讲, Selector 选择器就是"一个表示特殊语意的字符串"。 只要把选择器字符串传入上面的方法中就能够选择不同的 Dom 对象并且以 jQuery 包装集的形式返回。

jQuery 选择器按照功能主要分为"选择"和"过滤"。 并且是配合使用的，具体分类如下。基础选择器掌握即可 ,其他用到再查阅。

3.1. 基础选择器

选择器	名称	举例
id选择器	#id	$("#testDiv")选择id为testDiv的元素
元素名称选择器	element	$("div")选择所有div元素
类选择器	.class	$(".blue")选择所有class=blue的元素
选择所有元素	*	$("*")选择页面所有元素
组合选择器	selector1,selector2,selectorN	$("#testDiv,span,.blue")同时选中多个选择器匹配的元素

```
<style type="text/css">
  .blue{
    background: blue;
  }
</style>

<body>
  <div id="mydiv1">id选择器1<span>span中的内容</span></div>
  <div id="mydiv2" class="blue">元素选择器</div>
  <span class="blue">样式选择器</span>
</body>

<script src="js/jquery-3.4.1.js" type="text/javascript" charset="utf-8"></script>
<script type="text/javascript">
  // id选择器
  console.log("======id====");
  var idSelecter = $('#mydiv1');
  console.log(idSelecter.html());
  console.log(idSelecter.text());
  // 元素选择器
  console.log("======name====");
  var nameSe = $('div'); // 有多个div元素
  nameSe.each(function(){
    // this是dom对象, $(this)是jquery对象
    console.log($(this).text());
  });
  // 类选择器, class
```

```
  console.log("======class====");
  var classSe = $('.blue'); // 有多个class=blue的元素
  classSe.each(function(){
    console.log($(this).text());
  });
  // 通用选择器: *
   console.log("======所有元素====");
   var all = $("*");
   console.log(all.length);
  // 组合选择器
  console.log("======组合====");
  var unionSe = $('span, .blue,div');
  unionSe.each(function(){
    console.log($(this).text());
  });
</script>
```

3.2. 层次选择器

选择器	名称	举例
后代选择器	ancestor descendant	$("#parent div")选择id为parent的元素的所有div元素
子代选择器	parent > child	$("#parent>div")选择id为parent的直接div子元素
相邻选择器	prev + next	$(".blue + img")选择css类为blue的下一个img元素
同辈选择器	prev ~ sibling	$(".blue ~ img")选择css类为blue的之后的img元素

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>层次选择器</title>
    <script src="js/jquery-3.4.1.js" type="text/javascript"></script>
    <style type="text/css">
      .testColor{
        background: green;
      }
      .gray{
        background: gray;
      }
    </style>
  </head>
  <body>
    <div id="parent">层次择器
      <div id="child" class="testColor">父选择器
        <div class="gray">子选择器</div>
        <img src="http://www.baidu.com/img/bd_logo1.png"
```

```
                    width="270" height="129" />
        <img src="http://www.baidu.com/img/bd_logo1.png"
                    width="270" height="129" />
      </div>
      <div>
        选择器2<div>选择器2中的div</div>
      </div>
    </div>
  </body>
  <script type="text/javascript">
    console.log("==========后代选择器-选择所有后代=====");
    var ancestorS = $('#parent div');
    ancestorS.each(function(){
      console.log($(this).text());
    });

    console.log("==========子代选择器-选择儿子辈=====");
    var child = $('#parent>div');
    child.each(function(){
      console.log($(this).text());
    });

    console.log("==========相邻选择器=====");
    var pre_next = $(".gray + img");
    console.log(pre_next.length);

    console.log("==========同辈选择器,其后, (弟弟) =====");
    var pre_siblings = $(".gray ~ img");
    console.log(pre_siblings.length);
  </script>
</html>
```

3.3. 表单选择器

Forms	名称	举例
表单选择器	:input	查找所有的input元素：$(":input"); 注意：会匹配所有的input、textarea、select和button元素。
文本框选择器	:text	查找所有文本框：$(":text")
密码框选择器	:password	查找所有密码框：$(":password")
单选按钮选择器	:radio	查找所有单选按钮：$(":radio")
复选框选择器	:checkbox	查找所有复选框：$(":checkbox")
提交按钮选择器	:submit	查找所有提交按钮：$(":submit")
图像域选择器	:image	查找所有图像域：$(":image")
重置按钮选择器	:reset	查找所有重置按钮：$(":reset")
按钮选择器	:button	查找所有按钮：$(":button")
文件域选择器	:file	查找所有文件域：$(":file")

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>表单验证</title>
    <script src="js/jquery-3.4.1.js" type="text/javascript"></script>
  </head>
  <body>
    <form id='myform' name="myform"  method="post">
      <input type="hidden" name="uno" value="9999" disabled="disabled"/>
      姓名:<input type="text" id="uname" name="uname" /><br />
      密码:<input type="password" id="upwd" name="upwd" value="123456" /><br />
      年龄:<input type="radio" name="uage" value="0" checked="checked"/>小屁孩
         <input type="radio" name="uage" value="1"/>你懂得  <br />
      爱好:<input type="checkbox" name="ufav" value="篮球"/>篮球
         <input type="checkbox" name="ufav" value="爬床"/>爬床
         <input type="checkbox" name="ufav" value="代码"/>代码<br />
      来自:<select id="ufrom" name="ufrom">
          <option value="-1" selected="selected">请选择</option>
          <option value="0">北京</option>
          <option value="1">上海</option>
        </select><br />
      简介:<textarea rows="10" cols="30" name="uintro"></textarea><br />
      头像:<input type="file"  /><br />
      <input type="image" src="http://www.baidu.com/img/bd_logo1.png"
                  width="20" height="20"/>
      <button type="submit" onclick="return checkForm();">提交</button>
      <button type="reset" >重置</button>

    </form>
```

```
  </body>
</html>
<script type="text/javascript">
  function checkForm(){
    // 获取 所有的表单元素
    $(":input").each(function(){
      // console.log($(this)[0]);
      console.log($(this)[0].tagName);
    })
    console.log("------++++++++++++++++++++++++--------")
    // 获取 text
    console.log("text-->" + $(":text").length); // 1
    // 获取 password
    console.log("password-->" + $(":password").length); // 1
    // 获取radio
    console.log("radio-->" + $(":radio").length); // 2
    // 获取checkbox
    console.log("checkbox-->" + $(":checkbox").length); // 3
    // 获取file
    console.log("file-->" + $(":file").length); // 1
    // 获取按钮
    console.log("button-->" + $(":button").length); // 2
    // 获取submit按钮
    console.log("submit-->" + $(":submit").length); // 1
    // 获取image按钮
    console.log("image-->" + $(":image").length); // 1
    // 获取reset按钮
    console.log("reset-->" + $(":reset").length); // 1
    return false;
  }
</script>
```

4. Jquery Dom操作

jQuery也提供了对HTML节点的操作，而且在原生js的基础之上进行了优化，使用起来更加方便。

常用的从几个方面来操作，查找元素（选择器已经实现）；创建节点对象；访问和设置节点对象的值，以及属性；添加节点；删除节点；删除、添加、修改、设定节点的CSS样式等。注意：以下的操作方式只适用于jQuery对象。

4.1. 操作元素的属性

4.1.1. 获取属性

方法	说明	举例
attr(属性名称)	获取指定的属性值，操作 checkbox 时， 选中返回 checked，没有选中返回 undefined。	attr('checked') attr('name')
prop(属性名称)	获取具有true和false两个属性的属性值	prop('checked')

```
<form action="" id="myform">
  <input type="checkbox" name="ch" checked="checked"/> aa
  <input type="checkbox" name="ch" /> bb
</form>

<script type="text/javascript">
  var ch = $("input[type='checkbox']")
  console.log(ch)
  ch.each(function(idx, em){
      console.log(idx + "-" + $(em) + "=" + this)
      console.log($(em).attr('checked') + "==" + $(em).prop('checked'))
      console.log('--------------')
  })
</script>
```

4.1.2. 设置属性

方法	说明	举例
attr(属性名称，属性值)	设置指定的属性值， 操作 checkbox 时， 选中返回 checked， 没有选中返回 undefined。	attr('checked','checked') attr('name','zs')
prop(属性名称，属性值)	设置具有true和false的属性值	prop('checked','true')

4.1.3. 移除属性

方法	说明	举例
removeAttr(属性名)	移除指定的属性	removeAttr('checked')

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>属性操作</title>
    <script src="js/jquery-3.4.1.js" type="text/javascript"></script>
  </head>
  <body>
    <pre>
      <h5>1.attr()</h5>
```

```
        设置或者返回元素的属性 ；
        <h5>2.prop()</h5>
        设置 具有 true 和 false 两个属性的属性，如 checked，selected 或者 disabled。
      </pre>
      <hr />
      <a href="http://www.baidu.com" id="a1">百度</a>
      <a href="http://www.sina.com" id="a2">新浪</a>
          <input type="checkbox" name="all" checked="checked"/>全选
    </body>
    <script type="text/javascript">
      // 获取属性值：attr
      console.log($('#a1').attr('href'));
      console.log($(':checkbox').attr('name'));
      // 若未选中显示undefined，选中显示 checked
      console.log($(':checkbox').attr('checked'));
      // 获取属性值：prop
      // 若未选中显示false，选中显示 true
      console.log($(":checkbox").prop('checked'));
      console.log($('#a2').prop('href'))
      // 设置属性值
      $('#a1').attr('href','https://jquery.com');
      $(":checkbox").prop("checked",false);
      // 移除属性
      $('#a2').removeAttr('href');
    </script>
</html>
```

4.2. 操作元素的样式

对于元素的样式，也是一种属性，由于样式用得特别多，所以对于样式除了当做属性处理外还可以有专门的方法进行处理。

方法	说明
attr("class")	获取class属性的值，即样式名称
attr("class","样式名")	修改class属性的值，修改样式
addClass("样式名")	添加样式名称
css()	添加具体的样式
removeClass(class)	移除样式名称

增加元素的具体样式，格式：

1) css({‘样式名’:’样式值’,’样式名2’:’样式值2’})

例: css({"background-color":"red","color":"#fff"});

2) css(“样式名”,”样式值”)

例: css('color','white')

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>设置元素样式</title>
    <script src="jquery-3.4.1.js" type="text/javascript" ></script>
    <style type="text/css">
      div{
          padding: 8px;
          width: 180px;
      }
      .blue{
        background: blue;
      }
      .larger{
        font-size: 30px;
      }
      .green {
        background : green;
      }
    </style>
  </head>
  <body>
    <h3>css()方法设置元素样式</h3>
    <div id="conBlue" class="blue larger">天蓝色</div>
      <div id="conRed">大红色</div>
      <div id="remove" class="blue larger">天蓝色</div>
  </body>
  <script type="text/javascript">
    // 获取样式名称
    console.log($("#remove").attr("class"));
    // 修改样式, 那么id为remove的元素样式class只有green
    // $('#remove').attr("class","green")
    // 添加样式名称, class名称 --叠加
    // $('#conBlue').addClass("blue larger");
    // 添加元素具体样式
    // { "":"" , "":"" }   名:值     对
    $('#conRed').css({"background-color":"red","color":"#fff"});
    $('#remove').css('color','red');
    // 移除样式
    // $("#remove").removeClass("blue larger");
  </script>
</html>
```

4.3. 操作元素的内容

对于元素还可以操作其中的内容，例如文本，值，甚至是html。

方法	说明
html()	获取元素的html内容
html("html, 内容")	设定元素的html内容
text()	获取元素的文本内容，不包含html
text("text 内容")	设置元素的文本内容，不包含html
val()	获取元素value值
val("值")	设定元素的value值

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>操作内容</title>
    <script src="js/jquery-3.4.1.js" type="text/javascript"></script>
  </head>
  <body>
    <h3><span>html()和text()方法设置元素内容</span></h3>
    <div id="html"></div>
      <div id="text"></div>
        <input type="text" name="uname" value="oop" />
  </body>
  <script type="text/javascript">
    // 获取HTML内容，包括HTML标签
    console.log($('h3').html());
    // 获取文本内容，不包括HTML标签
    console.log($('h3').text());
    // 获取value值
    console.log($('[name=uname]').val());
    // 设置
    $('#html').html("<p>使用html设置,看不到标签</p>");
    $('#text').text("<p>使用text设置,能看到标签</p>");
    $('[name=uname]').val("哈哈哈");
    // console.info("abc");
    // console.log("abc");
    // console.warn("abc")
    // console.error("abc");
  </script>
</html>
```

4.4. 创建元素

在jQuery中创建元素很简单，直接使用核心函数即可

```
$('元素内容');
```

```
$('<p>this is a paragraph!!!</p>');
```

4.5. 添加元素

方法	说明
prepend(content)	在被选元素内部的开头插入元素或内容，被追加的 content 参数，可以是字符、HTML 元素标记。
$(content).prependTo(selector)	把 content 元素或内容加入 selector 元素开头
append(content)	在被选元素内部的结尾插入元素或内容，被追加的 content 参数，可以是字符、HTML 元素标记。
$(content).appendTo(selector)	把 content元素或内容插入selector 元素内，默认是在尾部
before()	在元素前插入指定的元素或内容:$(selector).before(content)
after()	在元素后插入指定的元素或内容:$(selector).after(content)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>追加</title>
    <script src="jquery-3.4.1.js" type="text/javascript" charset="utf-8"></script>
    <style type="text/css">
      div {
          margin: 10px 0px;
      }
      span{
          color: white;
          padding: 8px
      }
      .red{
          background-color: red;
      }
      .blue{
          background-color: blue;
      }
      .green{
          background-color: green;
      }
```

```
    </style>
  </head>
  <body>
    <h3>prepend()方法前追加内容</h3>
    <h3>prependTo()方法前追加内容</h3>
    <h3>append()方法后追加内容</h3>
    <h3>appendTo()方法后追加内容</h3>
    <span class="red">男神</span>
    <span class="blue">偶像</span>
    <div class="green">
            <span >小鲜肉</span>
        </div>
  </body>
</html>
<script type="text/javascript">
  var str ="<span id='mydiv' style='padding: 8px;width: 180px;background-
color:#ADFF2F;'>动态创建span</span>";
  // 1、使用prepend前加内容
  $("body").prepend(str);

  // 2、使用prependTo前加内容
  $("<b>开头</b>").prependTo('body');

  // 3、使用append后加内容
  $("body").append(str);
  // $("div").append($('.red')); // 当把已存在的元素添加到另一处的时候相当于移动

  // 4、使用appendTo后追加内容
  $(str).appendTo('body');
  // $('.blue').appendTo("div");
</script>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>插入元素</title>
    <script src="js/jquery-3.4.1.js" type="text/javascript" charset="utf-8"></script>
    <style type="text/css">
      span{
          color: white;
          padding: 8px
      }
      .red{
          background-color: red;
      }
      .blue{
          background-color: blue;
      }
      .green{
          background-color: green;
      }
```

```
    </style>
  </head>
  <body>
    <h3>before() 和 after()方法在元素之前后插入内容</h3>
        <span class="green">财大气粗</span>
  </body>
</html>
<script type="text/javascript">
   var str1 = "<span class='red'>土豪</span>";
   var str2 = "<span class='blue'>暴发户</span>";
   $(".green").before(str1); // 前置元素
   $(".green").after(str2); // 后存元素
</script>
```

4.6. 删除元素

方法	说明
remove()	删除所选元素或指定的子元素，包括整个标签和内容一起删。
empty()	清空所选元素的内容

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>删除元素</title>
    <script src="js/jquery-3.4.1.js" type="text/javascript"></script>
    <style type="text/css">
      span{
          color: white;
          padding: 8px;
          margin: 5px;
          float: left;
      }
      .green{
          background-color: green;
      }
      .blue{
          background-color: blue;
      }
    </style>
  </head>
  <body>
    <h3>删除元素</h3>
    <span class="green">jquery<a>删除</a></span>
        <span class="blue">javase</span>
        <span class="green">http协议</span>
        <span class="blue">servlet</span>
```

```
  </body>
</html>
<script type="text/javascript">
  // 删除所选元素 或指定的子元素
  // $("span").remove();
  // 删除样式为blue的span
  // $("span.blue").remove();
  // 清空元素
  // $("span").empty();
  // $(".green").empty();
</script>
```

4.7. 遍历元素

each()

$(selector).each(function(index,element)) :遍历元素

参数 function 为遍历时的回调函数，

index 为遍历元素的序列号，从 0 开始。

element是当前的元素，此时是dom元素。

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>遍历元素</title>
    <style type="text/css">
      span{
          color: white;
          padding: 8px;
          margin: 5px;
          float: left;
      }
      .green{
          background-color: green;
      }
      .blue{
          background-color: blue;
      }
    </style>
    <script src="jquery-3.4.1.js" type="text/javascript" charset="utf-8"></script>
  </head>
  <body>
    <h3>遍历元素 each()</h3>
    <span class="green">jquery</span>
        <span class="green">javase</span>
        <span class="green">http协议</span>
        <span class="green">servlet</span>
```

```
  </body>
  <script type="text/javascript">
    $('span').each(function (idx , e) {
      console.log(idx + " ---> " + $(e).text());
    })
  </script>
</html>
```

5. Jquery事件

5.1. ready加载事件

ready()类似于 onLoad()事件

ready()可以写多个，按顺序执行

$(document).ready(function(){})等价于$(function(){})

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>ready事件</title>
    <script src="js/jquery-3.4.1.js" type="text/javascript"></script>
    <script type="text/javascript">
      // 文档载入完便触发ready方法
      $(document).ready(function(){
        $("div").html("ready go...");
      })
      // $(document).ready(function(){}) == $(function(){})
      $(function(){
        $("p").click( function () {
          $(this).hide();
        });
      });
      $(function(){
        $("#btntest").bind("click",function(){
          $("div").html("剁吧...");
        });
      });
    </script>
  </head>
  <body>
    <h3>页面载入时触发ready()事件</h3>
    <div></div>
    <input id="btntest" type="button" value="剁手" />
    <p>aaa</p>
    <p>bbbb</p>
```

```
    <p>ccc</p>
    <p>dddd</p>
  </body>
</html>
```

5.2. bind()绑定事件

为被选元素添加一个或多个事件处理程序，并规定事件发生时运行的函数。

```
$(selector).bind( eventType [, eventData], handler(eventObject));
```

eventType： 是一个字符串类型的事件类型，就是你所需要绑定的事件。

这类类型可以包括如下：

blur, focus, focusin, focusout, load, resize, scroll, unload, click, dblclick

mousedown, mouseup, mousemove, mouseover, mouseout, mouseenter

mouseleave,change, select, submit, keydown, keypress, keyup, error

[, eventData]：传递的参数，格式：{名:值,名2:值2}

handler(eventObject)：该事件触发执行的函数

5.2.1. 简单的bind()事件

```
<script type="text/javascript">
  $(function(){
    /*$("#test").bind("click",function(){
      alert("世界会向那些有目标和远见的人让路！！");
    });*/
    /*
     * js的事件绑定
      ele.onclick=function(){};
     * */
    // 等同于上面的放方法
    $("#test").click(function(){
      alert("世界会向那些有目标和远见的人让路！！");
    });
    /*
     1.确定为哪些元素绑定事件
      获取元素
     2.绑定什么事件（事件类型）
      第一个参数：事件的类型
     3.相应事件触发的，执行的操作
      第二个参数：函数
     * */
    $("#btntest").bind('click',function(){
      // $(this).attr('disabled',true);
      $(this).prop("disabled",true);
    })
```

```
  });
  </script>
<body>
  <h3>bind()方简单的绑定事件</h3>
  <div id="test" style="cursor:pointer">点击查看名言</div>
  <input id="btntest" type="button" value="点击就不可用了" />
</body>
```

5.2.2. 绑定多个事件

```
<script type="text/javascript">
  $(function(){
    // 绑定click 和 mouseout事件
    /*$("h3").bind('click mouseout',function(){
      console.log("绑多个事件");
    });*/
    // 链式编程
    $("h3").bind('click',function(){
      alert("链式编程1");
    }).bind('mouseout',function(){
      $("#slowDiv").show("slow");//让slowDiv显示
    });
    /*$("#test").click(function(){
      console.log("点击鼠标了....");
    }).mouseout(function () {
      console.log("移出鼠标了...");
    });*/
    $("#test").bind({
      click:function(){
        alert("链式编程1");
      },
      mouseout:function(){
        $("#slowDiv").show("slow");
      }
    });
  });
</script>
<body>
  <h3>bind()方法绑多个事件</h3>
  <div id="test" style="cursor:pointer">点击查看名言</div>
  <div id="slowDiv"style=" width:200px; height:200px; display:none; ">
    人之所以能，是相信能
  </div>
</body>
```

6. Jquery Ajax

6.1. $.ajax

jquery 调用 ajax 方法：

格式：$.ajax({});

参数：

type：请求方式 GET/POST

url：请求地址 url

async：是否异步，默认是 true 表示异步

data：发送到服务器的数据

dataType：预期服务器返回的数据类型

contentType：设置请求头

success：请求成功时调用此函数

error：请求失败时调用此函数

get请求

```
$.ajax({
  type:"get",
  url:"js/cuisine_area.json",
  async:true,
  success : function (msg) {
    var str = msg;
    console.log(str);
    $('div').append("<ul></ul>");
    for(var i=0; i<msg.prices.length;i++){
      $('ul').append("<li></li>");
        $('li').eq(i).text(msg.prices[i]);
    }
  },
  error : function (errMsg) {
    console.log(errMsg);
    $('div').html(errMsg.responseText);
  }
});
```

post请求

```
$.ajax({
  type:"post",
  data:"name=tom",
  url:"js/cuisine_area.json",
  contentType: "application/x-www-form-urlencoded",
  async:true,
  success : function (msg) {
    var str = msg;
    console.log(str);
```

```
    $('div').append("<ul></ul>");
    for(var i=0; i<msg.prices.length;i++){
      $('ul').append("<li></li>");
      $('li').eq(i).text(msg.prices[i]);
    }
  },
  error : function (errMsg) {
    console.log(errMsg);
    $('div').html(errMsg.responseText)
  }
});
```

6.2. $.get

这是一个简单的 GET 请求功能以取代复杂 $.ajax 。

请求成功时可调用回调函数。如果需要在出错时执行函数，请使用 $.ajax。

```
// 1.请求json文件，忽略返回值
$.get('js/cuisine_area.json');
```

```
// 2.请求json文件，传递参数，忽略返回值
$.get('js/cuisine_area.json',{name:"tom",age:100});
```

```
// 3.请求json文件,拿到返回值,请求成功后可拿到返回值
$.get('js/cuisine_area.json',function(data){
  console.log(data);
});
```

```
// 4.请求json文件,传递参数,拿到返回值
$.get('js/cuisine_area.json',{name:"tom",age:100},function(data){
  console.log(data);
});
```

6.3. $.post

这是一个简单的 POST 请求功能以取代复杂 $.ajax 。

请求成功时可调用回调函数。如果需要在出错时执行函数，请使用 $.ajax。

```
// 1.请求json文件，忽略返回值
$.post('../js/cuisine_area.json');
```

```
// 2.请求json文件，传递参数，忽略返回值
$.post('js/cuisine_area.json',{name:"tom",age:100});
```

```
// 3.请求json文件,拿到返回值,请求成功后可拿到返回值
$.post('js/cuisine_area.json',function(data){
  console.log(data);
});
```

```
// 4.请求json文件,传递参数,拿到返回值
$.post('js/cuisine_area.json',{name:"tom",age:100},function(data){
  console.log(data);
});
```

6.4. $.getJSON

表示请求返回的数据类型是JSON格式的ajax请求

```
$.getJSON('js/cuisine_area.json',{name:"tom",age:100},function(data){
  console.log(data); // 要求返回的数据格式是JSON格式
});
```