# 【jQuery：简单好用的JavaScript代码库 第一天】

## 主要内容

1. jQuery的引入
2. jQuery选择器的使用
3. jQuery完成DOM编程

## 学习目标

|  |  |
| --- | --- |
| **知识点** | **要求** |
| jQuery的定义和特点 | 熟悉 |
| jQuery的页面加载函数 | 熟练 |
| jQuery的基本选择器 | 熟练 |
| jQuery的属性选择器 | 熟悉 |
| jQuery的位置选择器 | 熟悉 |
| jQuery的表单选择器 | 熟悉 |
| jQuery操作属性和样式 | 熟悉 |
| jQuery操作文本和增删元素 | 熟练 |
| jQuery操作事件 | 熟练 |
| jQuery对象和DOM对象转换 | 熟练 |
| jQuery中的迭代遍历方式 | 熟练 |
| jQueryDOM综合案例开发 | 练习 |

## 一 jQuery引入

### jQuery定义和特点

**为什么使用jQuery ?**

原生使用JavaScript编码存在问题

1选择器功能弱

2 DOM操作繁琐之极

3浏览器兼容性不好

4动画功能弱

jQuery的优点

1强大的选择器

2出色的DOM封装

3出色的浏览器兼容性

4强大的动画功能

5体积小，压缩后只有100KB左右

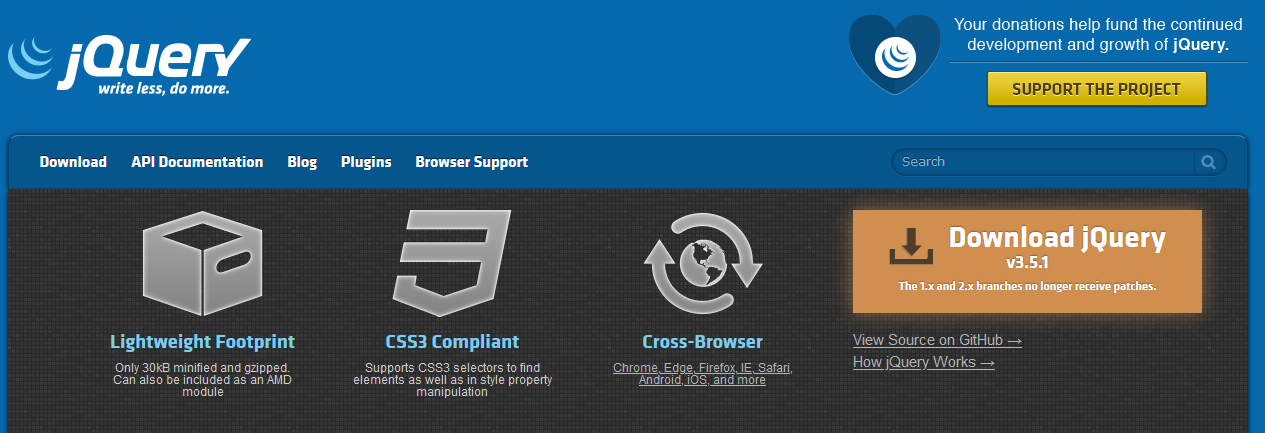
6可靠的事件处理机制

7使用隐式迭代简化编程

8丰富的插件支持

**什么是jQuery?**

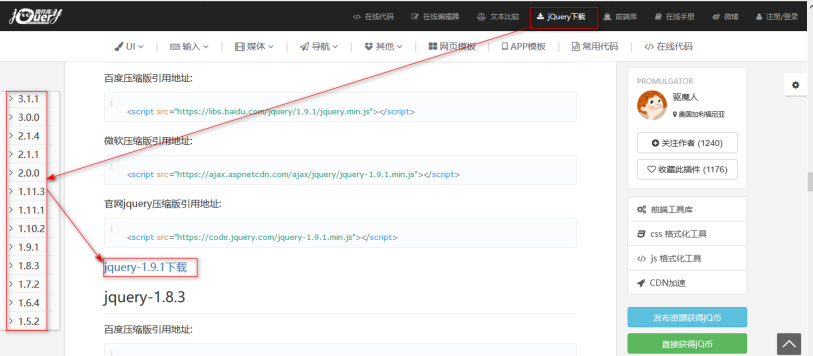
目前最流行的JavaScript函数库之一，对JavaScript进行了封装。并不是一门新语言。将常用的、复杂的操作进行函数化封装，直接调用，大大降低了使用JavaScript的难度，改变了使用JavaScript的习惯。jQuery能做的JavaScript也能做，但使用jQuery能大幅提高开发效率,由美国人John Resig在2006年推出，目前最新版本是v3.6.0。宗旨：**Write less，do more（写更少代码，做更多事情）**。官方网址<http://jquery.com/>



|  |
| --- |
| **官网关于jQuery的说明**  jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript. |
| **扩展:** **目前jQuery有三个大版本**   * 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。最新版本：3.6.0 |

**jQuery的下载**

jQuery中文网<http://www.jq22.com/>和官方网址<http://jquery.com/> 都可以提供下载



**jQuery如何引入?**

实现隔行变色的效果，在使用jQuery实现效果之前，先使用在HTML中使用class属性直接标记、使用JavaScript来实现动态隔行变色，然后再采用jQuery来实现，可以明显体现到jQuery的方便和简单。



原生JavaScript方式实现

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <style type="text/css">  .a {  background-color: greenyellow;  }    .b {  background-color: yellowgreen;  }    .c {  background-color: yellow;  }  </style>  <script type="text/javascript">  //页面加载后调用一个匿名函数  window.onload = function() {  //获取所有的行  var arr = document.getElementsByTagName("tr");  //对所有的行动态实现隔行变色  for(var i = 0; i < arr.length; i++) {  if(i == 0) {  arr[i].className = "a";  } else if(i % 2 == 1) {  arr[i].className = "b";  } else {  arr[i].className = "c";  }  }  }  </script>  </head>  <body>  <table class="datalist" summary="list of members in EE Studay" id="oTable">  <tr>  <th scope="col">Name</th>  <th scope="col">Class</th>  <th scope="col">Birthday</th>  <th scope="col">Constellation</th>  <th scope="col">Mobile</th>  </tr>  <tr>  <td>isaac</td>  <td>W13</td>  <td>Jun 24th</td>  <td>Cancer</td>  <td>1118159</td>  </tr>  <tr>  <td>fresheggs</td>  <td>W610</td>  <td>Nov 5th</td>  <td>Scorpio</td>  <td>1038818</td>  </tr>  <tr>  <td>girlwing</td>  <td>W210</td>  <td>Sep 16th</td>  <td>Virgo</td>  <td>1307994</td>  </tr>  <tr>  <td>tastestory</td>  <td>W15</td>  <td>Nov 29th</td>  <td>Sagittarius</td>  <td>1095245</td>  </tr>  </table>  </body>  </html> |

使用jQuery方式实现

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <!--引入jQuery文件-->  <script src="js/jquery.min.js"></script>  <script>  $(function(){  $("tr:odd").css("backgroundColor","yellowgreen")  $("tr:even").css("backgroundColor","lightyellow")  $("tr:first").css("backgroundColor","greenyellow")  })  </script>  </head>  <body>  <table class="datalist" summary="list of members in EE Studay" id="oTable">  <tr>  <th scope="col">Name</th>  <th scope="col">Class</th>  <th scope="col">Birthday</th>  <th scope="col">Constellation</th>  <th scope="col">Mobile</th>  </tr>  <tr>  <td>isaac</td>  <td>W13</td>  <td>Jun 24th</td>  <td>Cancer</td>  <td>1118159</td>  </tr>  <tr>  <td>fresheggs</td>  <td>W610</td>  <td>Nov 5th</td>  <td>Scorpio</td>  <td>1038818</td>  </tr>  <tr>  <td>girlwing</td>  <td>W210</td>  <td>Sep 16th</td>  <td>Virgo</td>  <td>1307994</td>  </tr>  <tr>  <td>tastestory</td>  <td>W15</td>  <td>Nov 29th</td>  <td>Sagittarius</td>  <td>1095245</td>  </tr>  </table>  </body>  </html> |

通过案例我们 发现,引入jQuery后我们的js代码变的简单,和write less,do more的目标特别契合

### jQuery页面加载函数

页面加载函数是页面加载完毕事件绑定的函数,该函数在后面应用比较多,所以再此处强调一下jQuery中的页面加载函数。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <!--引入jQuery的js文件-->  <script type="text/javascript" src="js/jquery.min.js" ></script>  <script>  /\*页面加载完毕后自动触发的方法  \* 通过绑定事件\*/  /\*function fun1(){  alert("页面加载完毕")  }\*/  /\*通过window对象 处理页面加载函数\*/    /\*window.onload=function (){  alert("页面加载完毕")  }\*/    /\*jQuery的核心函数\*/  /\*jQuery(document).ready(function (){  alert("页面加载完毕")  });\*/    /\*$(document).ready(function (){  alert("页面加载完毕")  });  \*/    $(function (){  alert("加载完毕")  })  </script>  </head>  <body ><!--onload="fun1()"-->  </body>  </html> |

**总结:**

关于jQuery的使用需要先导入jQuery的js文件,jQuery本身就是一个JS文件。jQuery.min.js是jQuery文件的压缩版,生产环境下推荐压缩版,$是jQuery是使用最多的符号，它有多个作用。这个示例中就使用了$的两个作用。

作用1：页面加载函数

jQuery(document).ready(function(){});和$(document).ready(function({}));简写为$(function(){}),相当于window.onload=function(){}但是功能比window.onload更强大window.onload一个页面只能写一个,但是可以写多个$() 而不冲突,window.onload要等整个页面加载完后再执行(包括图片、超链接、音视频等)，但是$()的执行时间要早

作用2：选择器标志 $(selector)

选择器。jQuery具有强大的选择器功能，后面会有专门章节进行介绍

### 本节作业

1 什么是jQuery?jQuery有什么优势?

1. jQuery中的页面加载函数和window.onload方式对比有什么优势?

## 二 jQuery选择器

jQuery提供了丰富的选择器功能，这个是jQuery相比JavaScript的一大优势。我们先来看一下jQuery API。可以看到提供了众多的选择器，可以非常方便简单的获取要选择的内容。

|  |
| --- |
| **问题：JavaScript是如何直接获取要选择的内容的**   * getElementById( ) ：返回一个节点对象 * getElementsByName( )：返回多个（节点数组） * getElementsByTagName( ) ：返回多个（节点数组）   jQuery的选择器功能要强大的多 |

### 2.1 基本选择器

标签选择器 $("a")

ID选择器 $("#id")     $("p#id")

类选择器 $(".class")    $("h2.class")

通配选择器 $("\*")

并集选择器$("elem1,elem2,elem3")

后代选择器$(ul li)

父子选择器 $(ul>li)

后面第一个兄弟元素 prev + next

后面所有的兄弟元素 prev ~ next

这些选择器其实我们并不陌生，因为在讲解CSS样式中都有类似的选择器，并且其含义也是一样的。不同的在CSS中是对选择的内容定义CSS样式，在jQuery中用来选择内容并后续进行更多的下步操作。

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title>基本选择器</title>  <style type="text/css">  .myClass{  background-color: aqua;  }  </style>  <script type="text/javascript" src="js/jquery.min.js" ></script>  <script type="text/javascript">  $(function(){  //标签选择器 $("a")  //$("h3").addClass("myClass");  //$("p").addClass("myClass");    //ID选择器 $("#id") $("p#id")  //$("#h31").addClass("myClass");  //$("h3#h31").addClass("myClass");    //类选择器 $(".class") $("h2.class")  //$(".red1").addClass("myClass");    //通配选择器 $("\*")  //$("\*").addClass("myClass");    //并集选择器$("elem1,elem2,elem3")  //$("h3,p,div").addClass("myClass");    //后代选择器$(ul li)  //$("p span").addClass("myClass");  //父子选择器 $(ul>li)  //$("p>span").addClass("myClass");  //后面第一个兄弟元素 prev + next  $("h3+p").addClass("myClass");  //后面所有的兄弟元素 prev ~ next  $("h3~p").addClass("myClass");  });  </script>  </head>  <body>  <h3 id="h31">JSP</h3>  <p>  JSP全名为Java Server Pages，中文名叫java服务器页面，其根本是一个简化的<span>Servlet</span>设计，  它[1] 是由Sun Microsystems公司倡导、许多公司参与一起建立的一种动态网页技术标准。JSP技术有点类似ASP技术，它是在传统的网页<em><span>HTML</span></em>（标准通用标记语言的子集）文件(\*.htm,\*.html) 中插入Java程序段(Scriptlet)和JSP标记(tag)，从而形成JSP文件，后缀名为(\*.jsp)。 用JSP开发的Web应用是跨平台的，既能在Linux下运行，也能在其他操作系统上运行。  </p>  <h3 id="h32" class="red1">Servlet</h3>  <p>  Servlet（Server Applet）是Java Servlet的简称，是为小服务程序或服务连接器，用Java编写的服务器端程序，主要功能在于交互式地浏览和修改数据，生成动态Web内容。  </p>  <p class="red1">  狭义的Servlet是指Java语言实现的一个接口，广义的Servlet是指任何实现了这个Servlet接口的类，一般情况下，人们将Servlet理解为后者。Servlet运行于支持Java的应用服务器中。从原理上讲，Servlet可以响应任何类型的请求，但绝大多数情况下Servlet只用来扩展基于HTTP协议的Web服务器。  </p>  <div>  <p>div p</p>  </div>  <span>span</span>  <p class="red1">  狭义的Servlet是指Java语言实现的一个接口，广义的Servlet是指任何实现了这个Servlet接口的类，一般情况下，人们将Servlet理解为后者。Servlet运行于支持Java的应用服务器中。从原理上讲，Servlet可以响应任何类型的请求，但绝大多数情况下Servlet只用来扩展基于HTTP协议的Web服务器。  </p>  </body>  </html> |

### 2.2 属性选择器

[attribute] 匹配包含给定属性的元素

[attribute1][attribute2] 复合属性选择器，需要同时满足多个属性

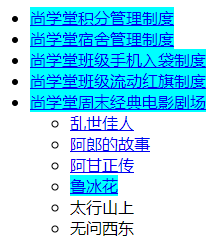
[attribute=value] 匹配给定的属性是某个特定值的元素

[attribute!=value] 匹配所有属性不等于特定值的元素

[attribute^=value] 匹配给定的属性是以某些值开始的元素

[attribute$=value] 匹配给定的属性是以某些值结尾的元素

[attribute\*=value] 匹配给定的属性是以包含某些值的元素



|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title>属性选择器</title>  <style type="text/css">  .myClass {  background-color: aqua;  }  </style>  <script src="js/jquery.js" type="text/javascript" charset="utf-8"></script>  <script type="text/javascript">  $(function() {  //[attribute]  //$("a").addClass("myClass");  //$("a[href]").addClass("myClass");  //[attribute1][attribute2]  //$("a[href][title]").addClass("myClass");  //[attribute=value]  //$("a[href='film-2.html']").addClass("myClass");  //[attribute!=value]  //$("a[href][href!='film-2.html']").addClass("myClass");  //[attribute^=value]  //$("a[href^='http']").addClass("myClass");  //[attribute$=value  //$("a[href$='htm']").addClass("myClass");  //[attribute\*=value]  $("a[href\*='bjsxt']").addClass("myClass");  });  </script>  </head>  <body>  <ul id="sxt">  <li>  <a href="http://www.bjsxt.com/score.html">尚学堂积分管理制度</a>  </li>  <li>  <a href="http://www.bjsxt.com/dorm">尚学堂宿舍管理制度</a>  </li>  <li>  <a href="http://www.bjsxt.com/mobile">尚学堂班级手机入袋制度</a>  </li>  <li>  <a href="http://www.bjsxt.com/flag">尚学堂班级流动红旗制度</a>  </li>  <li>  <a href="http://www.bjsxt.com/film">尚学堂周末经典电影剧场</a>  <ul id="film">  <li>  <a href="film-1.html">乱世佳人</a>  </li>  <li>  <a href="film-2.html" title="阿郎的故事">阿郎的故事</a>  </li>  <li id="film3">  <a href="film-3.html">阿甘正传</a>  </li>  <li>  <a href="http://www.bjsxt.com/film/film-4.htm" title="鲁冰花">  鲁冰花</a>  </li>  <li>  <a name="film-5.htm" title="太行山上">太行山上</a>  </li>  <li>无问西东</li>  </ul>  </li>  </ul>  </body>  </html> |

### 2.3 位置选择器

**针对整个页面而言的位置选择器**

:first 获取第一个元素

:last 获取最后一个元素

:odd 匹配所有索引值为奇数的元素，从 0 开始计数

:even匹配所有索引值为偶数的元素，从 0 开始计数

:eq(n) 匹配一个给定索引值的元素

:gt(n) 匹配所有大于给定索引值的元素

:lt(n) 匹配所有小于给定索引值的元素

**针对上级标签而言的位置选择器**

:first-child 匹配第一个子元素

:last-child匹配最后一个子元素

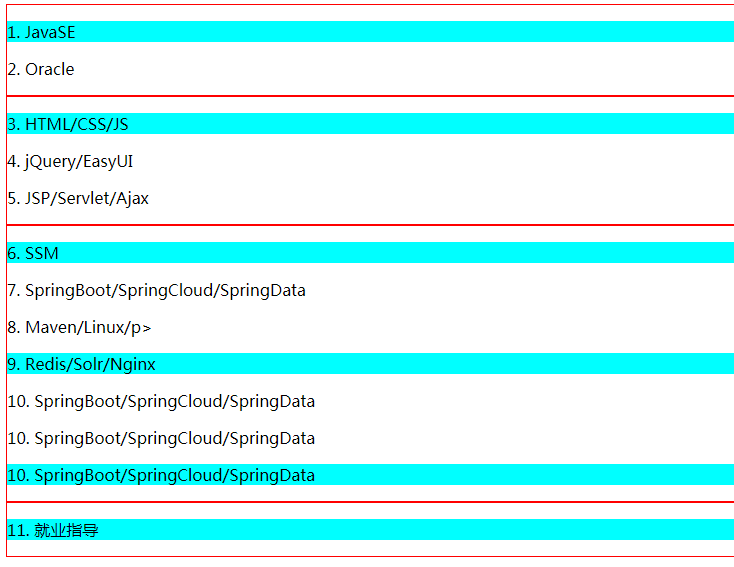
:only-child如果某个元素是父元素中唯一的子元素，将会被匹配

:nth-child(n) :nth-child(odd|even) :nth-child(xn+y) 匹配其父元素下的第N个子或奇偶元素

注意：nth-child()选择器编号是从

始，而其他选择器从0开始

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title>位置选择器</title>  <style type="text/css">  div{  border: 1px solid red;  }  .myClass{  background-color: aqua;  }  </style>  <script src="js/jquery.js" type="text/javascript" ></script>  <script type="text/javascript">  window.onload = function(){  //位置针对整个页面  //:first :last :odd :even  //$("p:first").addClass("myClass");  //$("p:last").addClass("myClass");  //$("p:odd").addClass("myClass");//索引从0开始  //$("p:even").addClass("myClass");    //:eq(n) :gt(n) :lt(n)  //$("p:eq(4)").addClass("myClass"); //equals  //$("p:lt(4)").addClass("myClass");//less than  //$("p:gt(4)").addClass("myClass");//greater than    //位置针对上级标签  //:first-child :last-child :only-child  //$("p:first-child").addClass("myClass");  //$("p:last-child").addClass("myClass");  //$("p:only-child").addClass("myClass");    //:nth-child(n) :nth-child(odd|even) :nth-child(xn+y)  //索引从0开始 只有此处从1开始  //$("p:nth-child(2)").addClass("myClass");  //$("p:nth-child(odd)").addClass("myClass");  //$("p:nth-child(even)").addClass("myClass");  $("p:nth-child(3n+1)").addClass("myClass");//n=0,1,2,3  }  </script>  </head>  <body>  <div>  <p>1. JavaSE</p>  <p>2. Oracle</p>  </div>  <div>  <p>3. HTML/CSS/JS</p>  <p>4. jQuery/EasyUI</p>  <p>5. JSP/Servlet/Ajax</p>  </div>  <div>  <p>6. SSM</p>  <p>7. SpringBoot/SpringCloud/SpringData</p>  <p>8. Maven/Linux/p>  <p>9. Redis/Solr/Nginx</p>  <p>10. SpringBoot/SpringCloud/SpringData</p>  <p>10. SpringBoot/SpringCloud/SpringData</p>  <p>10. SpringBoot/SpringCloud/SpringData</p>  </div>  <div>  <p>11. 就业指导</p>  </div>  </body>  </html> |



### 2.4 表单选择器

**关于表单项的选择器**

:text   :password  :radio  :checkbox  :hidden  :file  :submit

:input  匹配所有 input, textarea, select 和 button 元素

**关于表单项状态的选择器**

:selected  :checked  :enabled  :disabled :hidden :visible

**注意$("input")和$(":input")的区别**

$("input")：标签选择器，只匹配input标签,$(":input")： 匹配所有 input, textarea, select 和 button 元素

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title>表单选择器</title>  <style type="text/css">  .myClass {  background-color: aqua;  }  </style>  <script src="js/jquery.js" type="text/javascript"></script>  <script type="text/javascript">  $(function() {  //:text :password :radio :checkbox :hidden :file :submit  //$("input").addClass("myClass");  //var arr = $("input[type=hidden]");  //var arr = $("input:hidden");  //:input 匹配所有 input, textarea, select 和 button 元素  //var arr = $("input,select,textarea,button");  //var arr = $(":input");  //:selected :checked :enabled :disabled  //var arr = $(":disabled");  //var arr = $(":enabled");  //var arr = $(":input:not(:disabled)");  //var arr = $(":checked");  //var arr = $(":selected");  //:hidden :visible  var arr = $("input:hidden")  //var arr = $(":input:visible")  for(var i = 0; i < arr.length; i++) {  //alert(arr[i]);  console.info(arr[i]);  }  });  </script>  </head>  <body>  <h3>注册用户</h3>  <form action="doRegister.jsp" method="get">  <table border="1" width="40%">  <tr>  <td>用户名</td>  <td><input type="hidden" name="username" value="请输入姓名" />  <input type="text" name="username" id="username" value="请输入姓名" disabled="disabled" /></td>  </tr>  <tr>  <td>密 码</td>  <td><input type="password" name="pwd" id="pwd" /></td>  </tr>  <tr>  <td>确认密码</td>  <td><input type="color" name="repwd" id="repwd" /></td>  </tr>  <tr>  <td>性别</td>  <td>  <input type="radio" name="sex" value="男" />男  <input type="radio" name="sex" checked="checked" value="女" />女  </td>  </tr>  <tr>  <td>年龄</td>  <td><input type="text" min="6" max="30" name="age" id="age" value="18" /></td>  </tr>  <tr>  <td>电子邮箱</td>  <td><input type="text" name="email" id="email" /></td>  </tr>  </table>  </form>  </body>  </html> |

### 本节作业

1练习基本选择器

2练习属性选择器

3练习位置选择器

4练习表单选择器

## 三 jQuery完成DOM编程

### 3.1 jQuery操作属性和样式

**操作属性**

原生js 中的通过元素.属性名或者元素.setAttribute()方式操作元素属性,jQuery给我们封装了attr() 和removeAttr(),更加便捷的操作属性

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title></title>  <style>  .a{  background-color: #ADFF2F;  }  </style>  <script type="text/javascript" src="js/jquery.min.js" ></script>  <script>  function fun1(){  /\*  attr() 传入属性名就是获取属性值  \* \*/  console.info($('#f1').attr("color"))  console.info($('#f1').attr("size"))  }  function fun2(){  /\*传入的是属性名和属性值 修改属性 如果没有该属性 那么就是增加属性\*/  $("#f1").attr("color","yellow")  $("#f1").attr("size","5")  }  function fun3(){  /\*根据属性名删除属性\*/  $("#f1").removeAttr("color")  }  function fun4(){  $("#f1").attr("class","a")  }  </script>  </head>  <body>  <font id='f1' color="red" size="7" >2020年的第一场雪</font>  <input type="button" value="获得属性" onclick="fun1()" />  <input type="button" value="修改属性" onclick="fun2()" />  <input type="button" value="删除属性" onclick="fun3()" />  <input type="button" value="添加属性" onclick="fun4()" />  </body>  </html> |

**操作样式**

原生js 中的通过元素.style.样式名=’样式值’的方式操作元素样式,jQuery给我们封装了css()方法,便于我们操作样式,多数情况样式选择器使用类选择器,所以jQuery针对于这一情况,给我们封装了addClass removeClass toggleClass 三个方法

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title></title>  <style>  .a{  width: 100px;  height: 100px;  background-color: pink;  }  .b{  width: 200px;  height: 200px;  background-color: yellow;  border: 1px solid green;  }  </style>  <script type="text/javascript" src="js/jquery.min.js" ></script>  <script>  function fun1(){  //获取样式css(样式名) 设置样式css(样式名,样式值)  $("#d1").css("background-color","deeppink")  $("#d1").css("width","150px")  $("#d1").css("height","150px")  console.info($('#d1').css("height"))  }  /\*样式往往和标签的class属性相关,所以我们可以通过操作标签的class属性处理样式  jQuery还为操作class属性提供了专门的API  \* \*/  function fun2(){  /\*给标签的class属性添加一个 b值\*/  $("#d2").addClass("b")  }  function fun3(){  /\*移除class属性中的b值\*/  $("#d2").removeClass("b")  }  function fun4(){  /\*切换class属性中 的b值 有则干掉,没有则添加\*/  $("#d2").toggleClass("b")  }  </script>  </head>  <body>  <div id="d1" class="a">    </div>  <input type="button" value="修改样式" onclick="fun1()" />  <div id="d2" >  d2  </div>      <input type="button" value="添加class" onclick="fun2()" />  <input type="button" value="删除class" onclick="fun3()" />  <input type="button" value="切换class" onclick="fun4()" />  </body>  </html> |

### 3.2 jQuery操作文本和增删元素

**操作文本**

原生js 中的通过元素.innerText和innerHTML和.value属性操作标签内部文本和内容,jQuery给我们封装了text(),html()和val()三个方法

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <style>  #d1{  width: 200px;  height: 200px;  border: 1px solid green;  }  </style>    <script type="text/javascript" src="js/jquery.min.js" ></script>  <script>  function fun1(){  console.info($('#d1').html());  console.info($('#d1').text());  console.info($('#i1').val())  }  function fun2(){  /\*$('#d1').html('<h3>三级标题</h3>');\*/  $('#d1').text('<h3>三级标题</h3>');  $('#i1').val('这是修改的内容');  }  function fun3(){  /\*$('#d1').html('');\*/  /\*$('#d1').text('');  $('#i1').val('');\*/  $('#d1').empty();  }  </script>  </head>  <body>  <input type="text" value="这里是文字" id='i1' />  <div id='d1'>  a  <span>xxx</span>  b  </div>  <input type="button" value="获得标签内容" onclick="fun1()"/>  <input type="button" value="修改标签内容" onclick="fun2()"/>  <input type="button" value="删除标签中的内容" onclick="fun3()"/>  </body>  </html> |

**增删元素**

原生js 中的对于元素的创建,增加和删除代码比较繁琐,而jQuery从元素的创建到元素的增加和删除都给我们提供了更加便捷的方法

创建元素

$('<span>text<span>')

追加元素

append() appendTo() 添加内部标签

before() insertBefore() 向前增加标签

after() insertAfter() 向后增加标签

删除元素

empty() 清空字标签

remove() 移除当前标签

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <style>  #d1{  width: 200px;  height: 200px;  border: 1px solid red;  }  </style>  <script type="text/javascript" src="js/jquery.min.js" ></script>  <script>  function fun1(){  /\*创建标签并设置样式\*/  var span1=$('<span>这里是文字<span>')  span1.css('color','green');  span1.css('border','1px solid blue')  /\*将创建的标签放到div中\*/  $('#d1').append(span1)  }  function fun2(){  /\*创建标签并设置样式\*/  var span1=$('<span>这里是文字<span>').css('color','green').css('border','1px solid blue')  /\*将创建的标签放到div中\*/  span1.appendTo($('#d1'))  }    function fun3(){  /\*创建标签并设置样式\*/  var span1=$('<span style="color:green;border:1px solid blue">这里是文字<span>')  /\*将新标签放入div之前\*/  $('#d1').before(span1)  }  function fun4(){  /\*创建标签并设置样式\*/  var span1=$('<span style="color:green;border:1px solid blue">这里是文字<span>')  /\*将新标签放入div之前\*/  span1.insertBefore($('#d1'))  }  function fun5(){  /\*创建标签并设置样式\*/  var span1=$('<span style="color:green;border:1px solid blue">这里是文字<span>')  /\*将新标签放入div之前\*/  $('#d1').after(span1)  }  function fun6(){  /\*创建标签并设置样式\*/  var span1=$('<span style="color:green;border:1px solid blue">这里是文字<span>')  /\*将新标签放入div之前\*/  span1.insertAfter($('#d1'))  }  function fun7(){  /\*删除标签中的所有子标签\*/  /\*$('#d1').empty();\*/  /\*删除当前元素本身\*/  $('#d1').remove();  }  </script>  </head>  <body>  <div id='d1'></div>  <input type="button" value="testAppend" onclick="fun1()" />  <input type="button" value="testAppendTo" onclick="fun2()" />  <input type="button" value="testbefore" onclick="fun3()" />  <input type="button" value="testinsertBefore" onclick="fun4()" />  <input type="button" value="testafter" onclick="fun5()" />  <input type="button" value="testInsertAfter" onclick="fun6()" />  <input type="button" value="empty" onclick="fun7()" />  </body>  </html> |

### 3.3 jQuery操作事件

**操作事件**

无非就是绑定事件,触发事件,解绑定事件.原生js中的通过DOM编程和在标签上的事件属性绑定事件,

jQuery中,我们可以使用

事件的绑定:bind(),live()(1.8及之前可用),on()(1.8之后推荐使用),one()

事件解绑定:unbind()

事件的触发:行为触发, jQuery方法触发

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title></title>  <style>  #d1{  width: 200px;  height: 200px;  border: 1px solid red;  }  </style>  <script type="text/javascript" src="js/jquery.min.js" ></script>  <script>  function fun1(){  /\*通过jQuery绑定事件\*/  /\*jQuery中的事件名是js中的事件名去掉前面的on onclick click \*/    // 调用bind方法动态给标签绑定事件  $('#d1').bind('mouseover',function (){  $('#d1').css("background-color","yellow")  })    $('#d1').bind('mouseleave',function (){  $('#d1').css("background-color","greenyellow")  })    // jQeury中针对没一个事件都有单独的事件绑定方法,方法名就是对应的事件名  /\*$('#d1').mouseover(function (){  $('#d1').css("background-color","yellow")  })    $('#d1').mouseleave(function (){  $('#d1').css("background-color","greenyellow")  })\*/    /\*绑定事件一次\*/  /\*$('#d1').one('click',function (){  $('#d1').css("background-color","yellow")  })    $('#d1').one('mouseleave',function (){  $('#d1').css("background-color","greenyellow")  })\*/    /\*  \*live 和on 方法他也是给节点绑定事件  \*为页面上后续新增的元素自动绑定事件  \*这两个方法功能一样 但是有版本差别  \* <=1.8 用live  \* >=1.8 用on  \* \*/  }  function fun2(){  /\*解除所有事件的绑定\*/  /\*$('#d1').unbind()\*/  /\*解除绑定指定事件\*/  $('#d1').unbind("mouseleave")  }    function fun3(){  /\*通过代码触发事件  就好比真的发生了什么行为  这个行为有没有绑定具体的事件和这里有没有发生事件是两回事  \*  \* \*/  $("#i1").focus()  $("#d1").mouseover();  }  function fun4(){  alert("获得焦点了")  }    </script>  </head>  <body>  <div id='d1' ></div>  <input type="button" value="添加事件" onclick="fun1()" />  <input type="button" value="解除绑定" onclick="fun2()" />  <br />  <input type="text" id='i1' onfocus="fun4()"/>  <input type="button" value="触发事件" onclick="fun3()" />  </body>  </html> |

### 3.4 jQuery对象和DOM对象的转换

使用原生JS方式获得的页面结点对象我们可以简称为DOM对象,使用jQuery核心函数获得的对象我们可以简称为jQuery对象,这两种方式获得的对象即是是页面上同一个元素,那么也是不一样的,二者之间的API是不通用的.而在某些情况下,我们往往无法选择接收的对象,只能被动使用,那么这个时候我们可以让二者实现转换,以达到可以调用API实现功能的目的

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script type="text/javascript" src="js/jquery.min.js" ></script>  <script>  $(function(){  /\*使用两种方式获得div中的文本\*/  var d1=document.getElementById('d1');  console.info(d1.innerText)    var $d1=$('#d1');  console.info($d1.text())    /\*为什么两种放式获得的对象方法不能通用?  \* 两种方式获得的对象其实是不一样的  \* js方式获得元素就是获得一个元素节点  \* jQuery方式获得的更像是一个数组 目标元素就是数组中的元素  \* \*/  /\*对比两种方式获得的对象差异\*/  console.info(d1)  console.info($d1)  /\*可以将两种方式获得的对象进行转换\*/  /\*将js对象转化为jQuery对象\*/  console.info( $(d1).text())  /\*将jQuery对象转换为js\*/  console.info($d1[0].innerText)  console.info($d1.get(0).innerText)    $('#d1').click(function (){  console.info($(this).text())  })    })  </script>    </head>  <body>  <div id='d1'>这里是文字</div>  </body>  </html> |

注意:

使用原生JSDOM对象转换成jQuery对象方式是$(dom对象),jQuery对象转换成DOM对象的方式是jQuery对象[0]/.get(0)

### 3.5 jQuery中的迭代遍历方式

jQuery给我们封装了一个快捷遍历元素的方法,接下来我们就使用一下jQuery中新的遍历方式

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title></title>  <script type="text/javascript" src="js/jquery.min.js" ></script>  <script>  $(function(){  var $lis =$('li')  console.info($lis)  for(var i = 0;i<$lis.length;i++){  /\*遍历出的每个元素是DOM对象\*/  console.info($lis[i].innerText)  }  for(var i in $lis){  console.info($lis[i].innerText)  }  /\*遍历所有元素的方法\*/  /\*  each每拿出一个元素 都会执行一次内部的function  i 当前元素的索引  e 当前元素 DOM对象  \*  \* \*/  $lis.each(function (i,e){  console.info(i+'>>>'+$(e).text())  })    $.each($lis,function (i,e){  console.info(i+'>>>'+$(e).text())  })  })  </script>  </head>  <body>  <ul>  <li>AI</li>  <li>Python</li>  <li>大数据</li>  <li>JAVA</li>  <li>前端</li>  </ul>  </body>  </html> |

### 本节作业

1 jQuery都可以完成那些DOM操作?使用的API都是什么?

1. jQuery对象和原生DOM对象之间的API能否互用?
2. 完成DOM阶段案例开发